6.S094: Deep Learning for Self-Driving Cars 2018

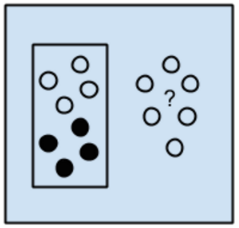
https://selfdrivingcars.mit.edu

Lex Fridman

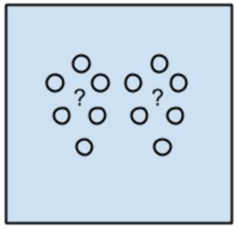
Lecture 4:

Computer Vision

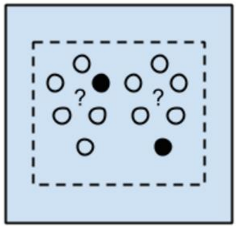
Computer Vision is Deep Learning



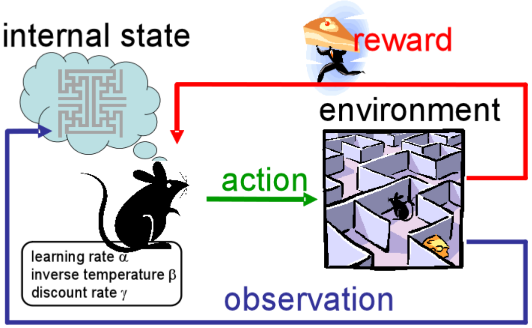
Supervised Learning



Unsupervised Learning

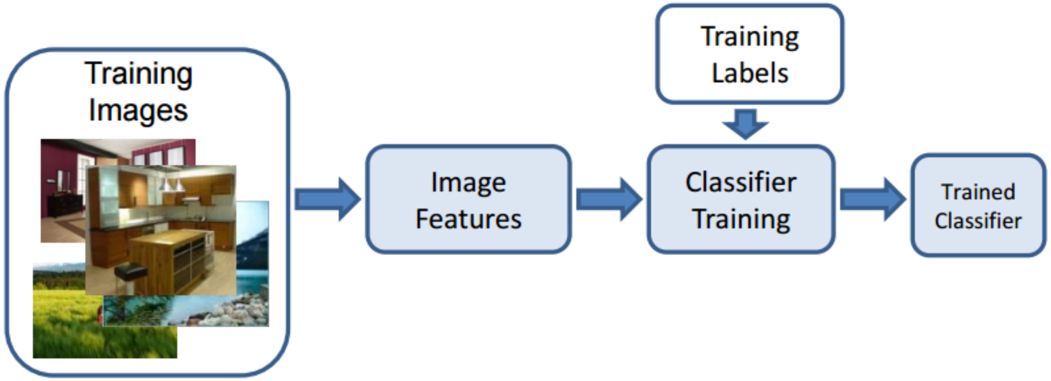


Semi-Supervised Learning

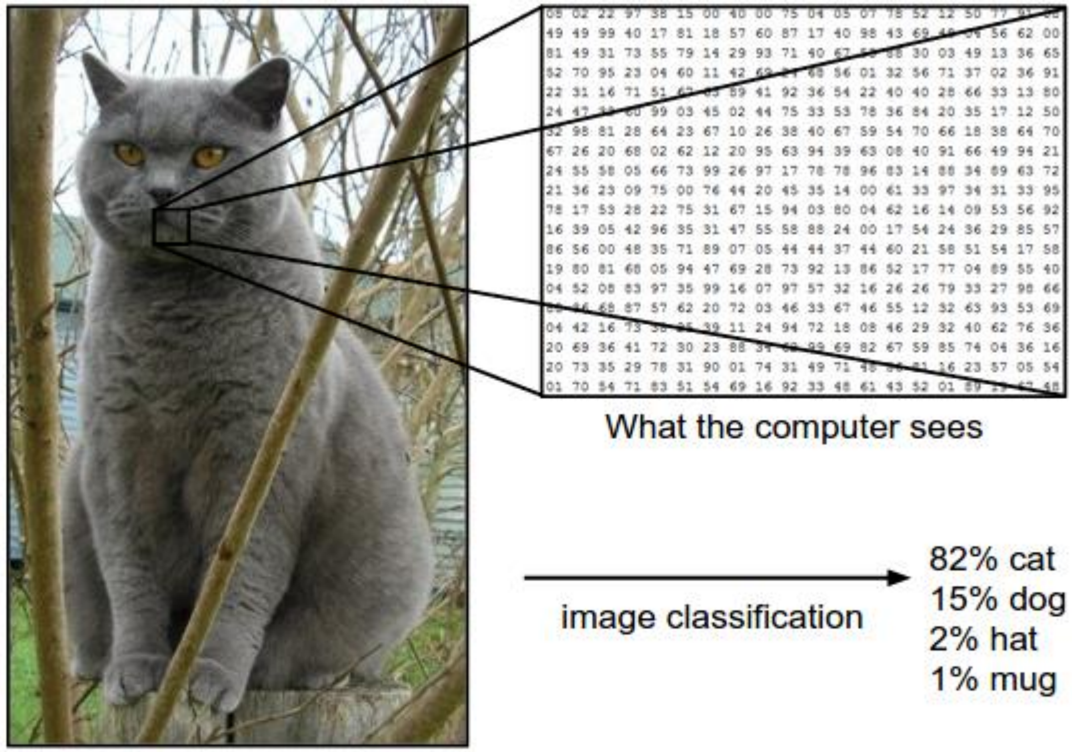


Reinforcement Learning

Computer Vision



Images are Numbers



- **Regression:** The output variable takes continuous values
- **Classification:** The output variable takes class labels
  - Underneath it may still produce continuous values such as probability of belonging to a particular class.

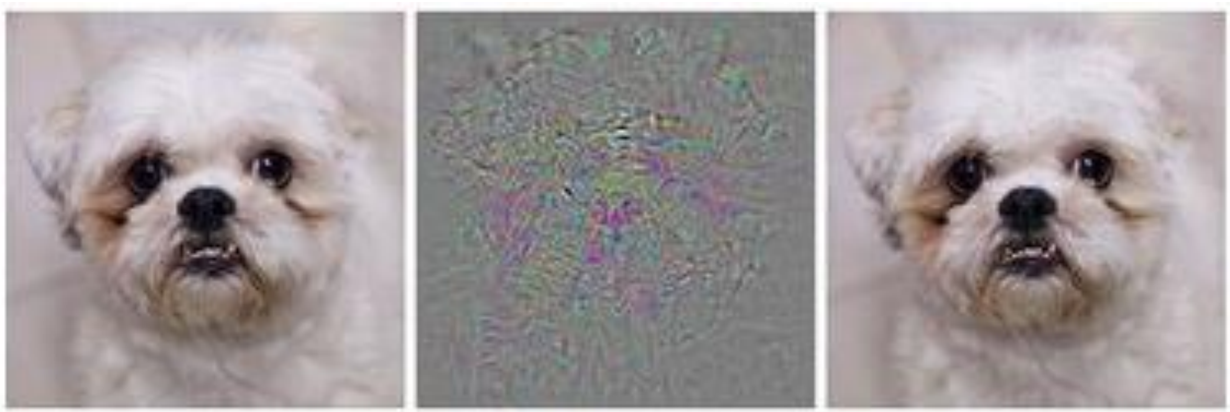
Computer Vision with Deep Learning:

Our intuition about what's "hard" is flawed (in complicated ways)

Visual perception: 540,000,000 years of data

Bipedal movement: 230,000,000 years of data

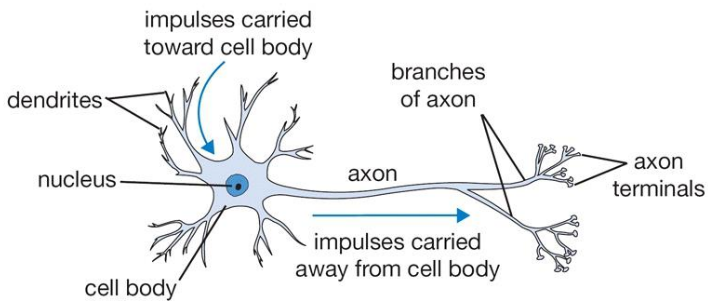
Abstract thought: 100,000 years of data



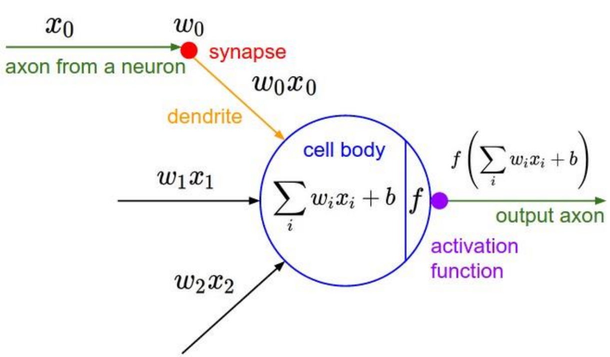
Prediction: **Dog** + Distortion Prediction: **Ostrich**

"Encoded in the large, highly evolve sensory and motor portions of the human brain is a **billion years of experience** about the nature of the world and how to survive in it.… Abstract thought, though, is a new trick, perhaps less than **100 thousand years** old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it."
- Hans Moravec, Mind Children (1988)

Neuron: Biological Inspiration for Computation



- **Neuron:** computational building block for the brain



- **(Artificial) Neuron:** computational building block for the "neural network"

Differences (among others):

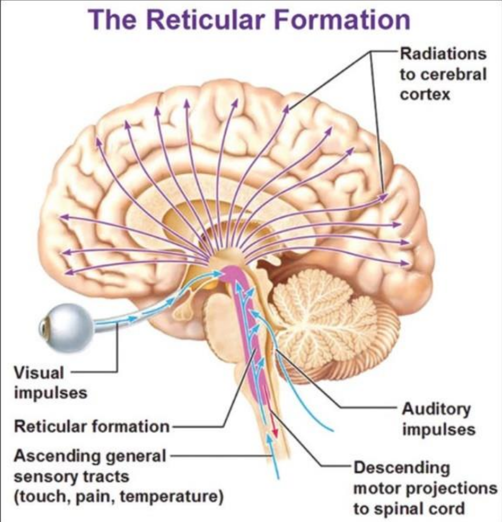
- **Parameters:** Human brains have ~10,000,000 times synapses than artificial neural networks.
- **Topology:** Human brains have no "layers". Topology is complicated.
- **Async:** The human brain works asynchronously, ANNs work synchronously.
- **Learning algorithm**: ANNs use gradient descent for learning. Human brains use … (we don't know)
- **Processing speed**: Single biological neurons are slow, while standard neurons in ANNs are fast.
- **Power consumption:** Biological neural networks use very little power compared to artificial networks
- **Stages:** Biological networks usually don't stop / start learning. ANNs have different fitting (train) and prediction (evaluate) phases.

Similarity (among others):

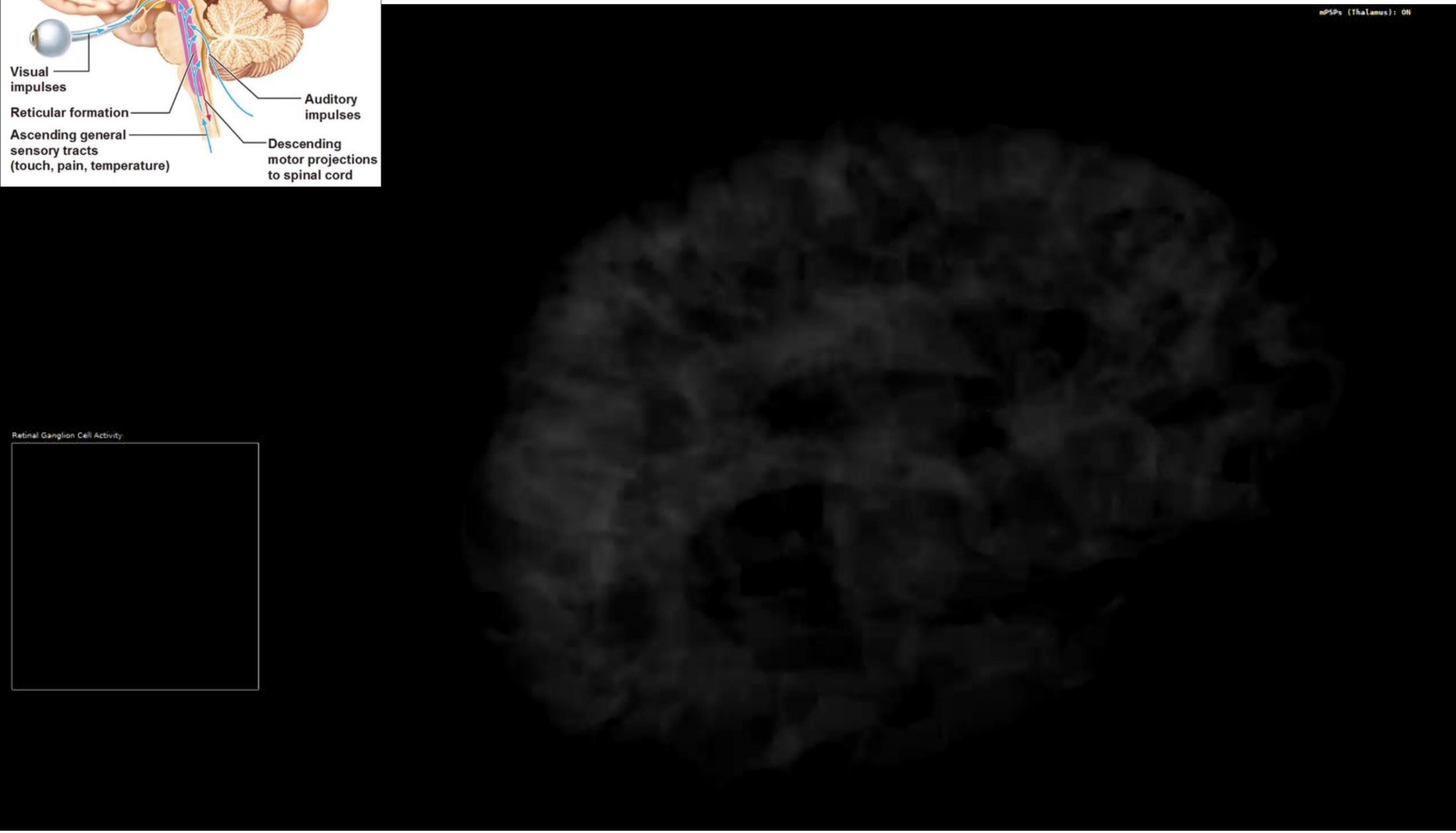
- Distributed computation on a large scale.

Human Vision

Its structure is instructive and inspiring!

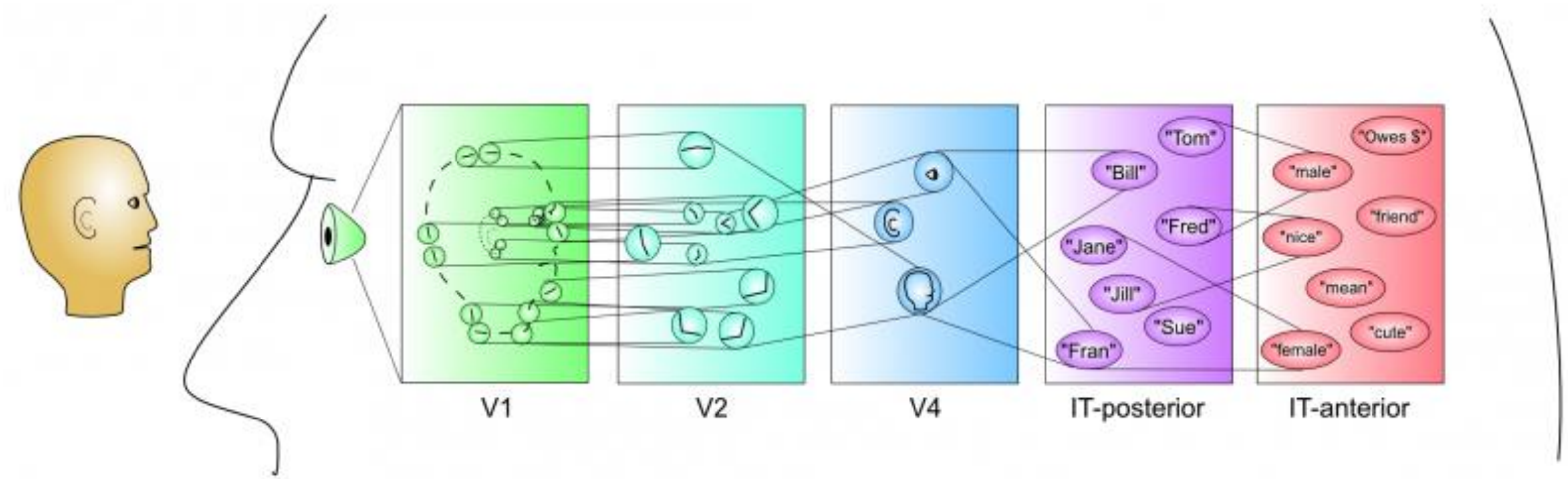


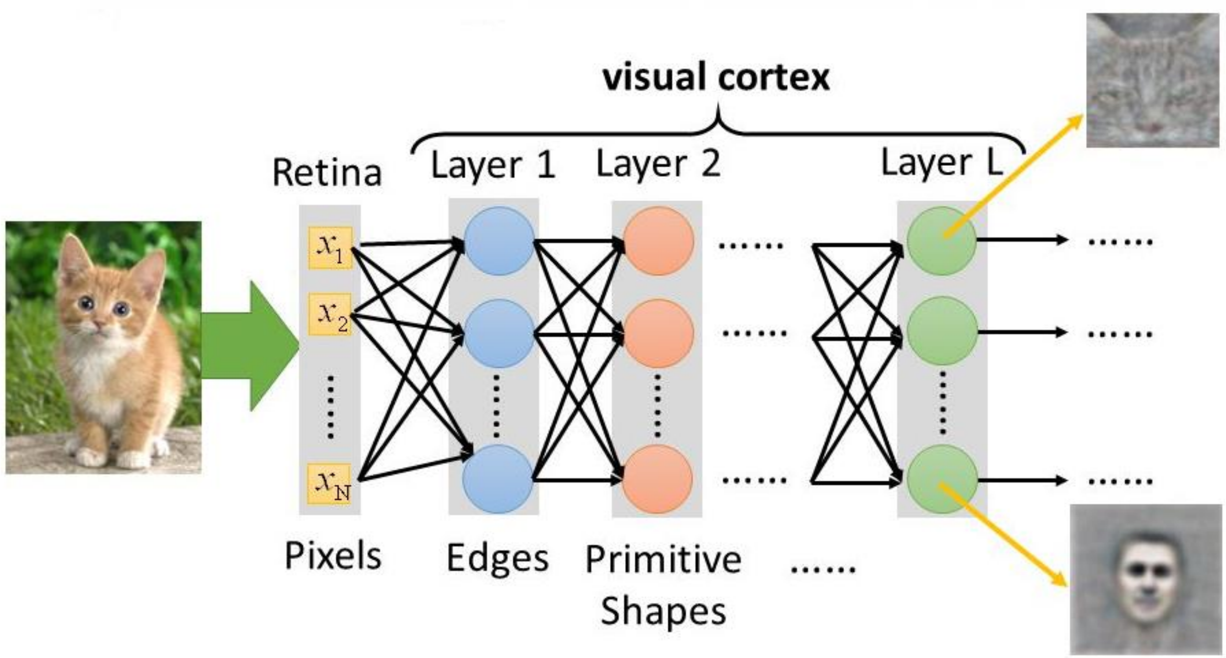
Thalamocortical System Simulation: 8 million cortical neurons + 2 billion synapses:



Visual Cortex

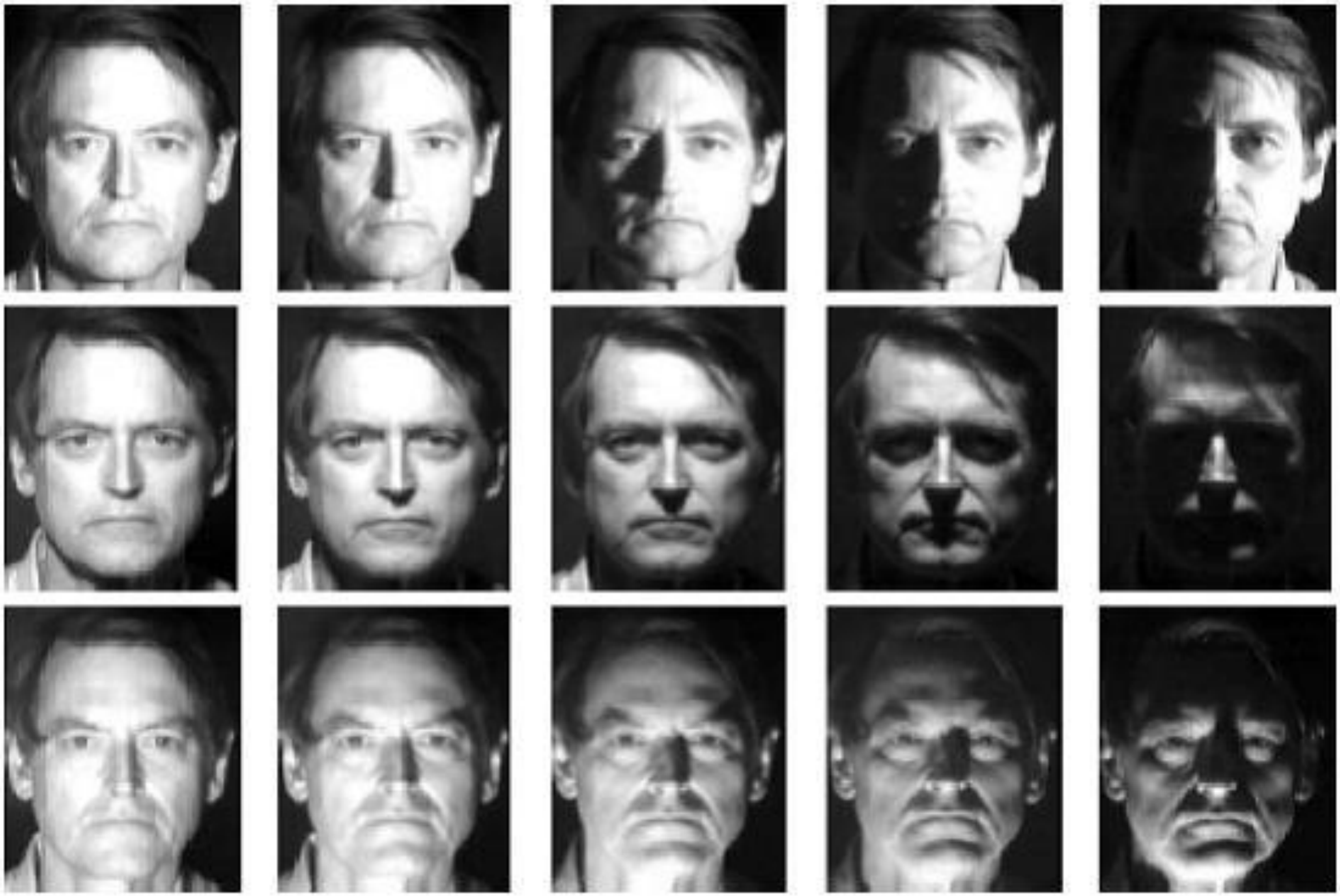
(Its Structure is Instructive and Inspiring)





5rjs.cn 专注无人驾驶

Deep Learning is Hard:
Illumination Variability



Deep Learning is Hard:

Pose Variability

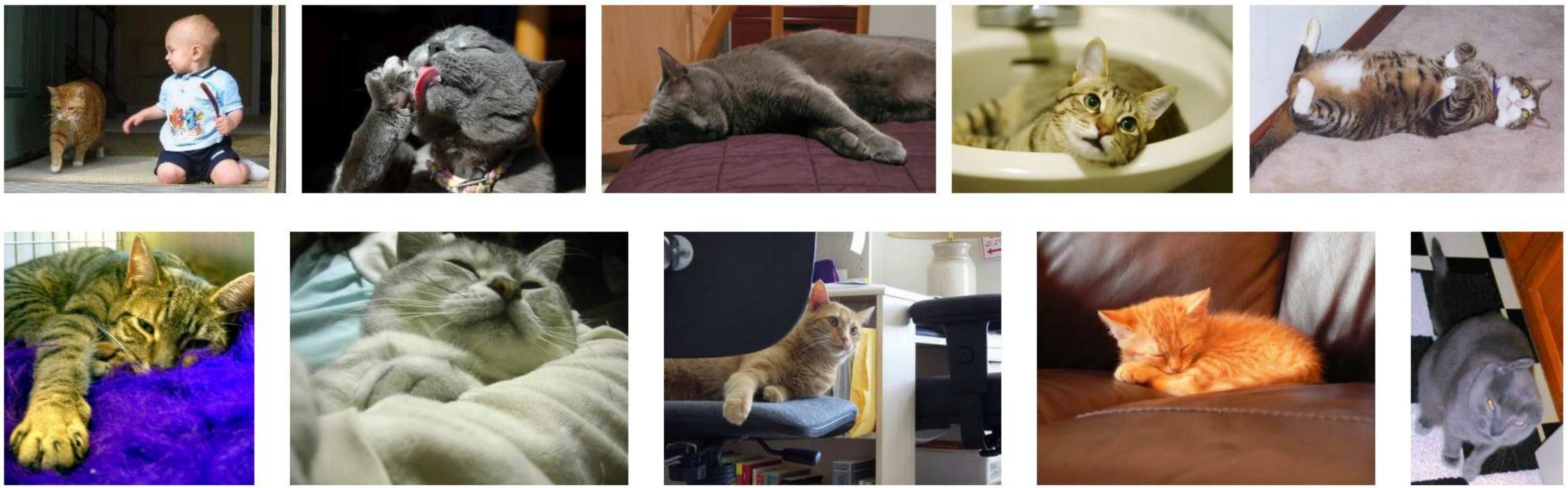
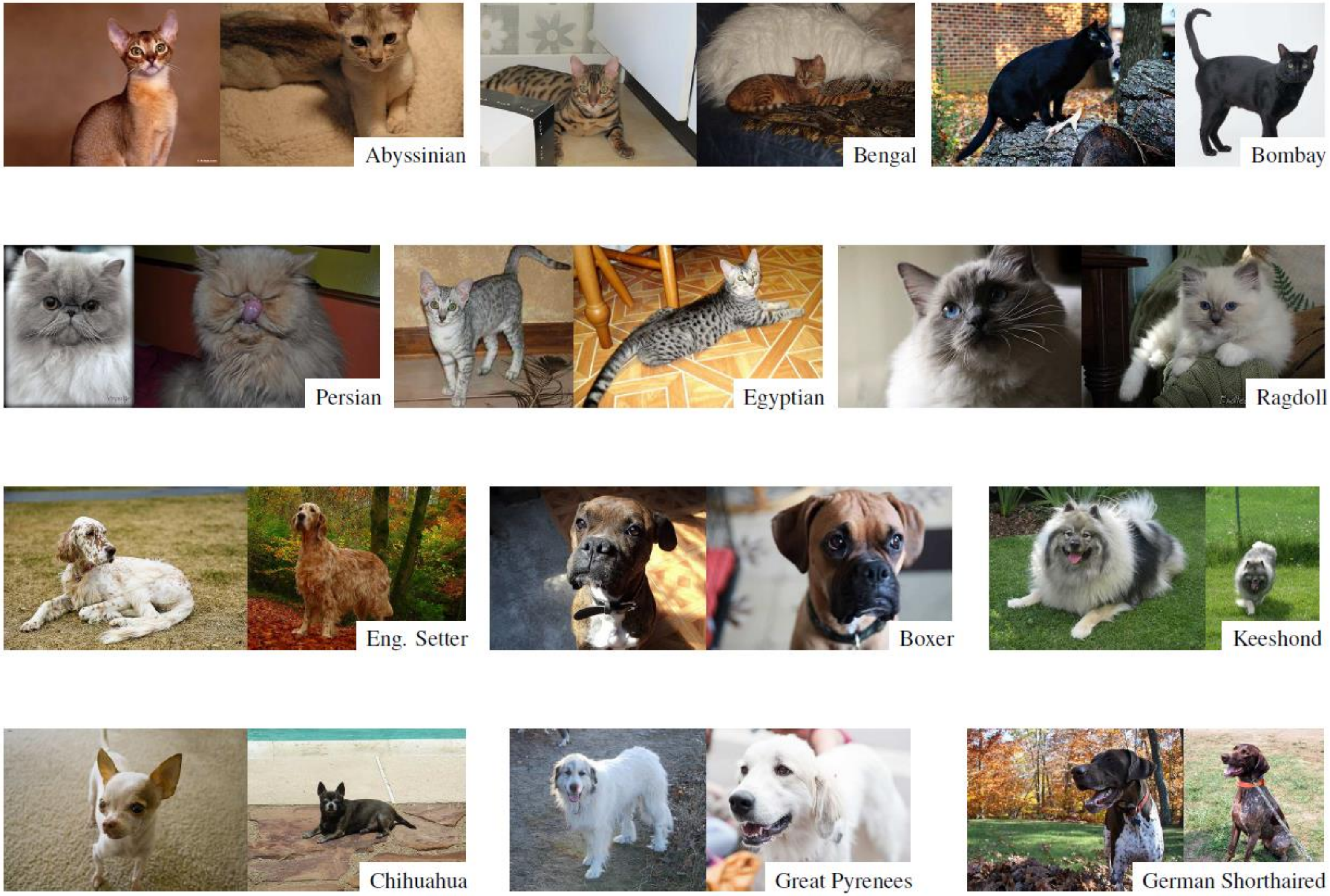


Figure 1. **The deformable and truncated cat.** Cats exhibit (al-

Parkhi et al. "The truth about cats and dogs.“ 2011.

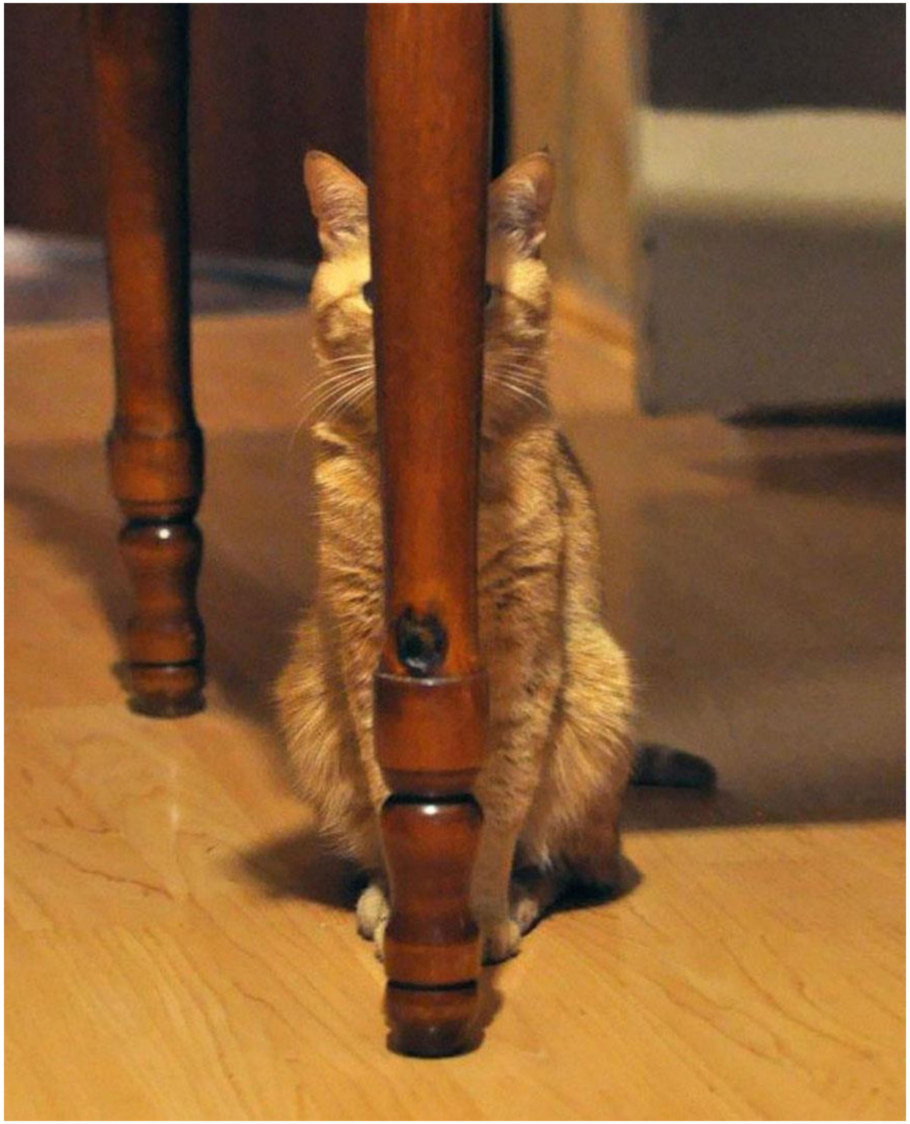
Deep Learning is Hard:

Intra-Class Variability



Parkhi et al. "Cats and dogs." 2012.

Occlusion

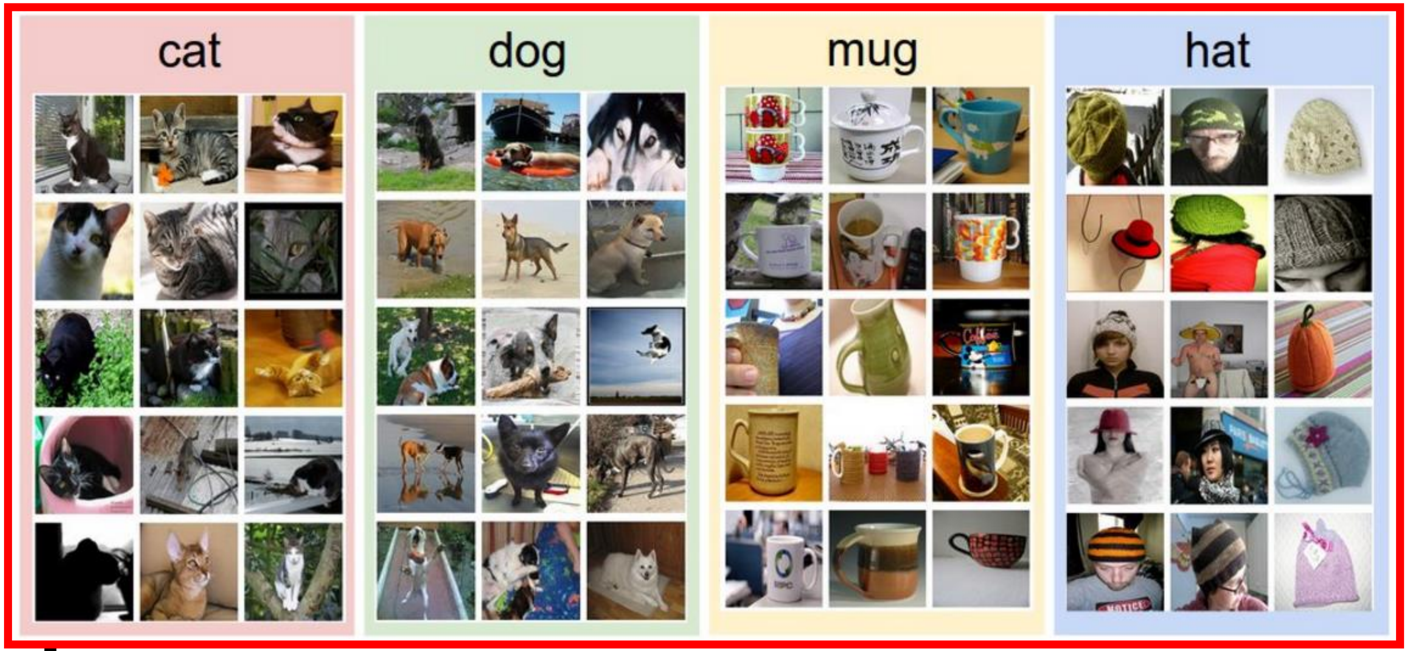


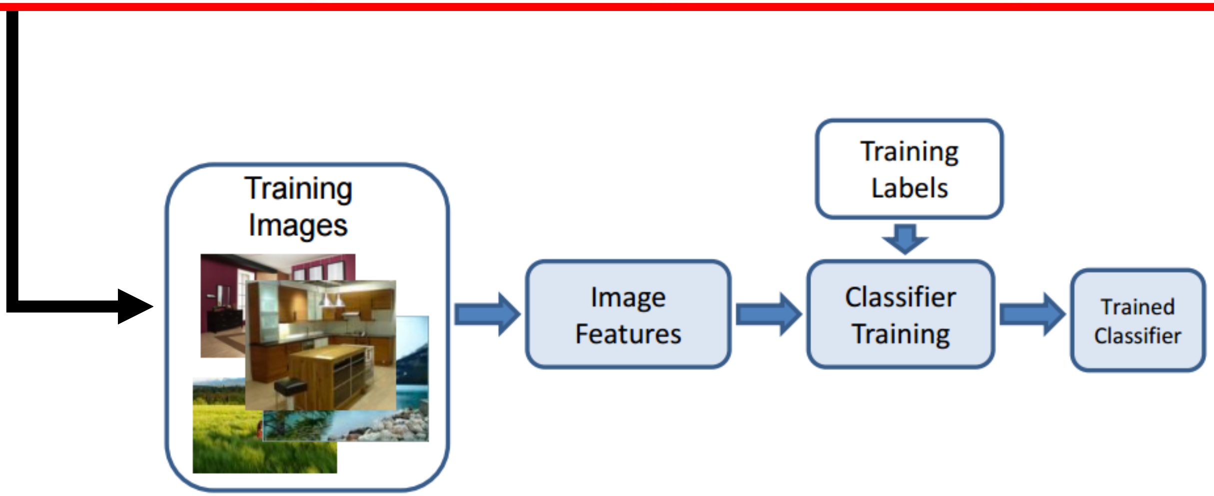
Occlusion

Occlusion

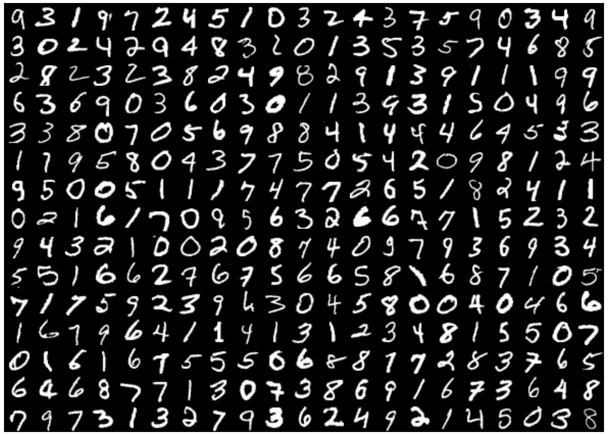
Philosophical Ambiguity: “Image Classification” is not (yet) “Understanding”

Image Classification Pipeline

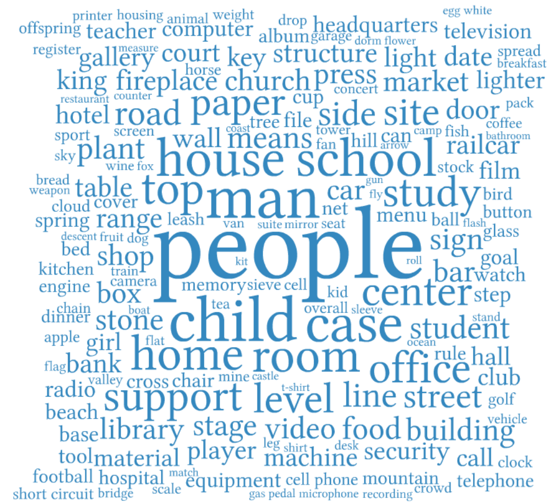




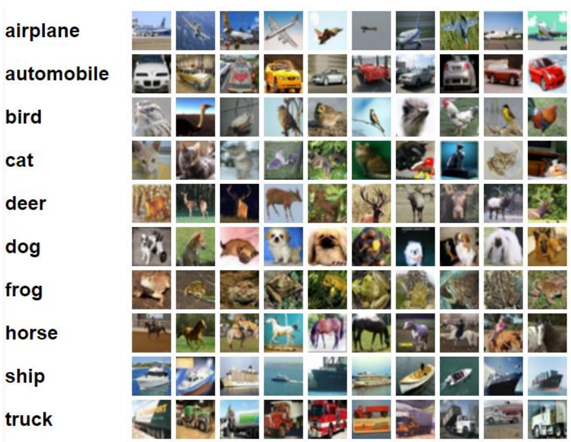
Famous Computer Vision Datasets



MNIST: handwritten digits



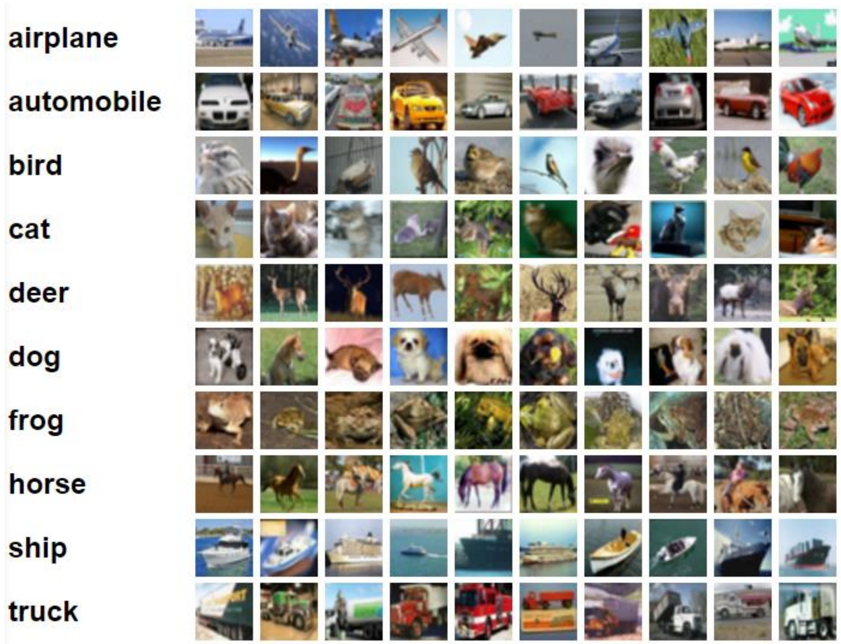
ImageNet: WordNet hierarchy



CIFAR-10(0): tiny images

Places: natural scenes

Let’s Build an Image Classifier for CIFAR-10



$$\left| \overset{\text{test image}}{\begin{bmatrix} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{bmatrix}} - \overset{\text{training image}}{\begin{bmatrix} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{bmatrix}} \right| = \overset{\text{pixel-wise absolute value differences}}{\begin{bmatrix} 46 & 12 & 14 & 1 \\ 82 & 13 & 39 & 33 \\ 12 & 10 & 0 & 30 \\ 2 & 32 & 22 & 108 \end{bmatrix}} \rightarrow 456$$

Let’s Build an Image Classifier for CIFAR-10

$$\left| \underset{\text{test image}}{\begin{array}{|c|c|c|c|}\hline 56 & 32 & 10 & 18 \\ \hline 90 & 23 & 128 & 133 \\ \hline 24 & 26 & 178 & 200 \\ \hline 2 & 0 & 255 & 220 \\ \hline \end{array}} - \underset{\text{training image}}{\begin{array}{|c|c|c|c|}\hline 10 & 20 & 24 & 17 \\ \hline 8 & 10 & 89 & 100 \\ \hline 12 & 16 & 178 & 170 \\ \hline 4 & 32 & 233 & 112 \\ \hline \end{array}} \right| = \underset{\text{pixel-wise absolute value differences}}{\begin{array}{|c|c|c|c|}\hline 46 & 12 & 14 & 1 \\ \hline 82 & 13 & 39 & 33 \\ \hline 12 & 10 & 0 & 30 \\ \hline 2 & 32 & 22 & 108 \\ \hline \end{array}} \rightarrow 456$$

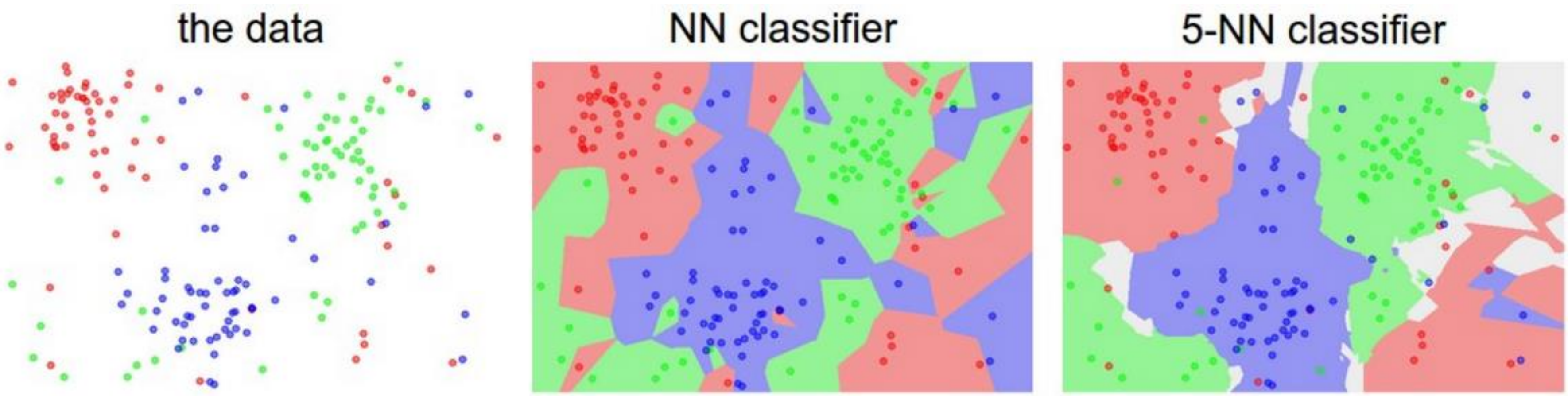
Accuracy

Random: **10%**

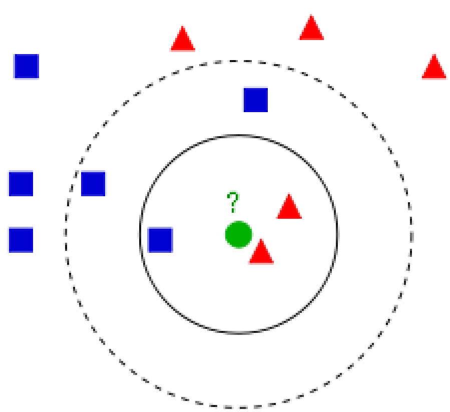
Our image-diff (with L1): **38.6%**

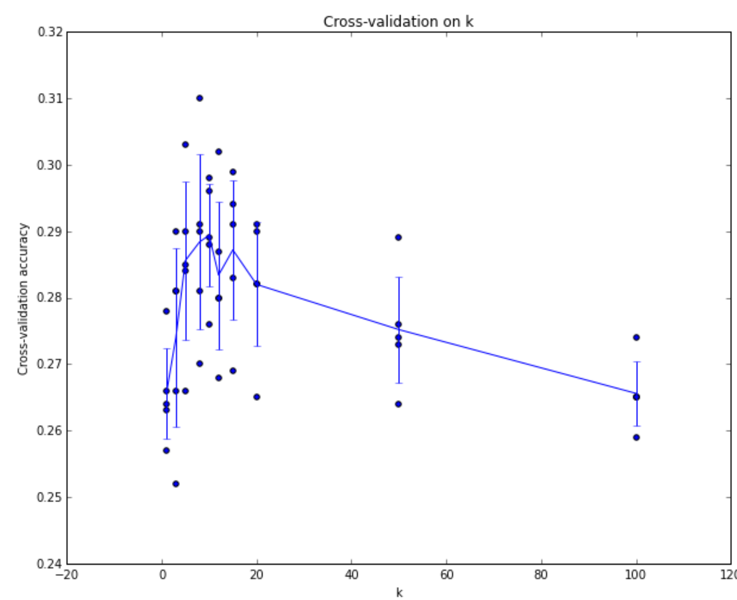
Our image-diff (with L2): **35.4%**

K-Nearest Neighbors: Generalizing the Image-Diff Classifier

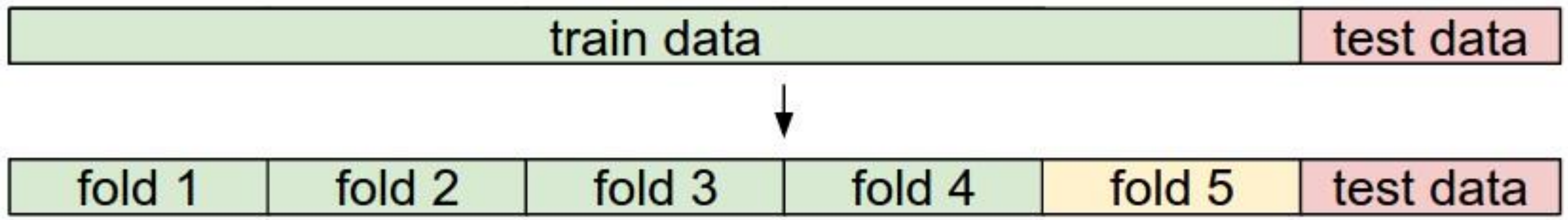


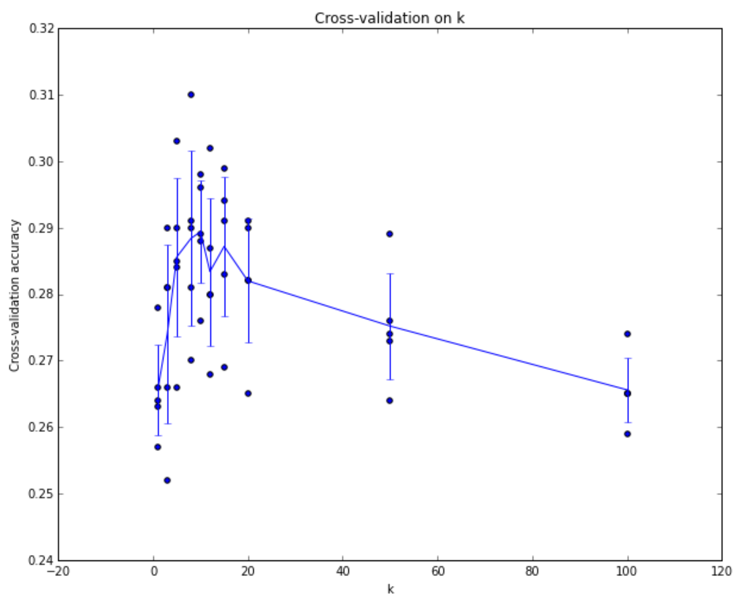
Tuning (hyper)parameters:





K-Nearest Neighbors: Generalizing the Image-Diff Classifier





Accuracy

Random: **10%**

Training and testing on the same data: **35.4%**

7-Nearest Neighbors: **~30%**

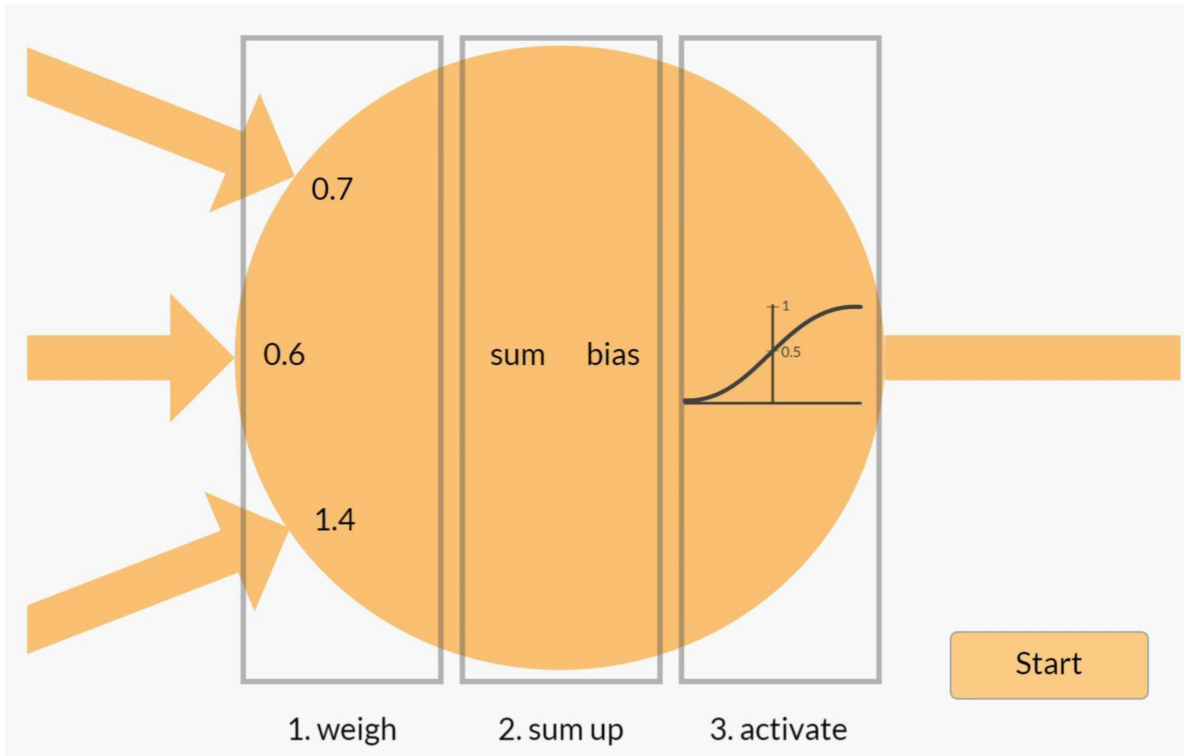
Human: **~95%**

…

Convolutional Neural Networks: **~97.75%**

Reminder: Weighing the Evidence

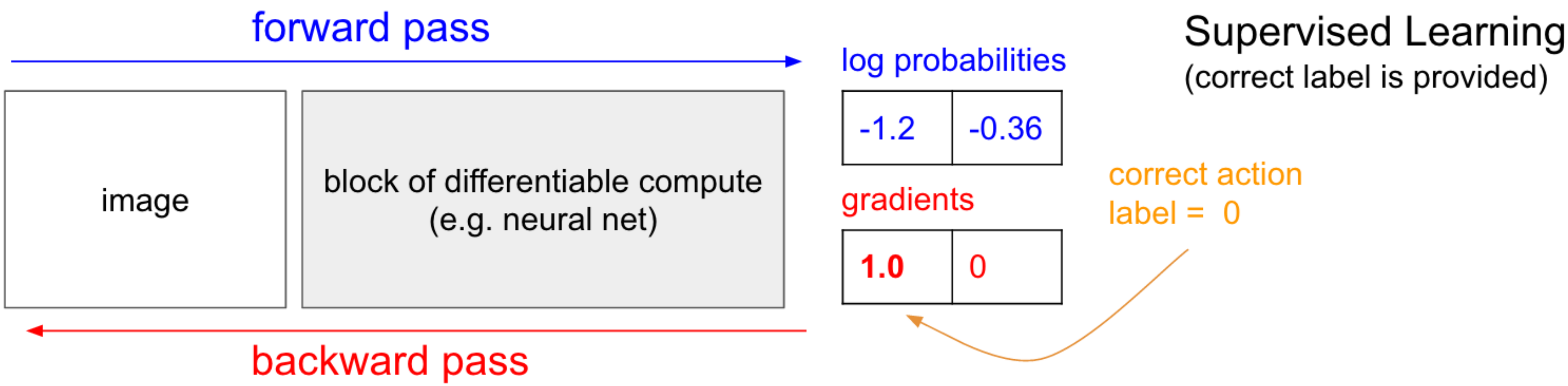
Evidence



Decisions

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Reminder: "Learning" is Optimization of a Function

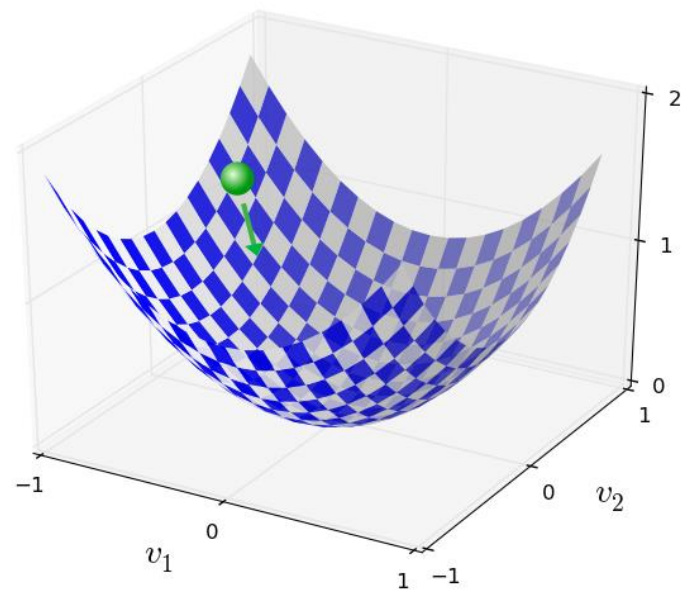


Ground truth for "6":

$$y(x) = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0)^T$$

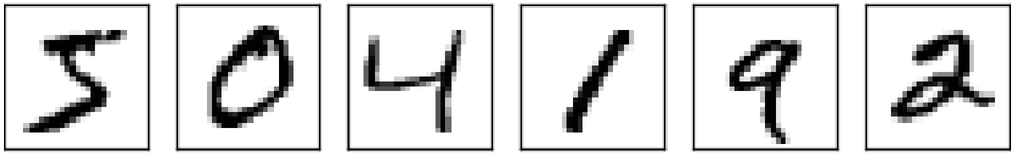
"Loss" function:

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

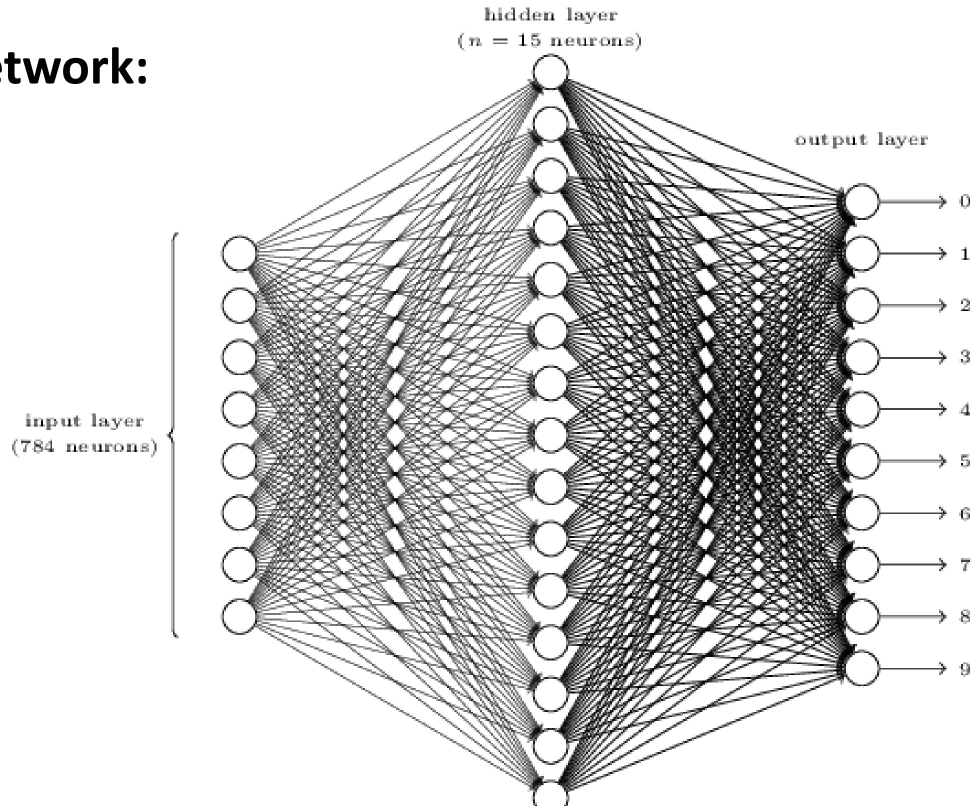


Classify and Image of a Number

Input:
(28x28)

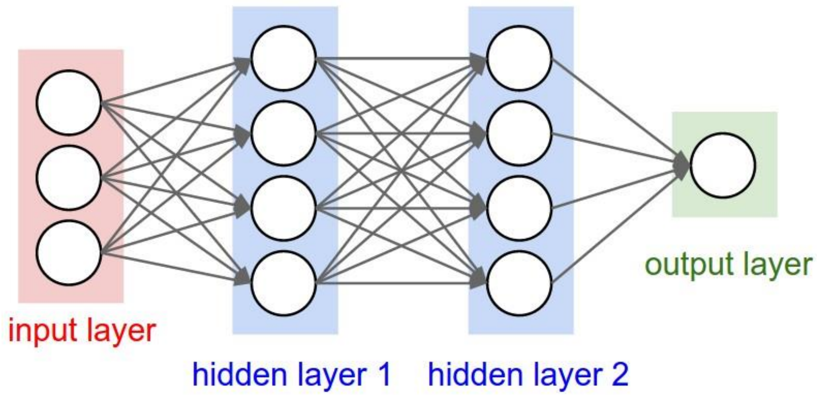


Network:

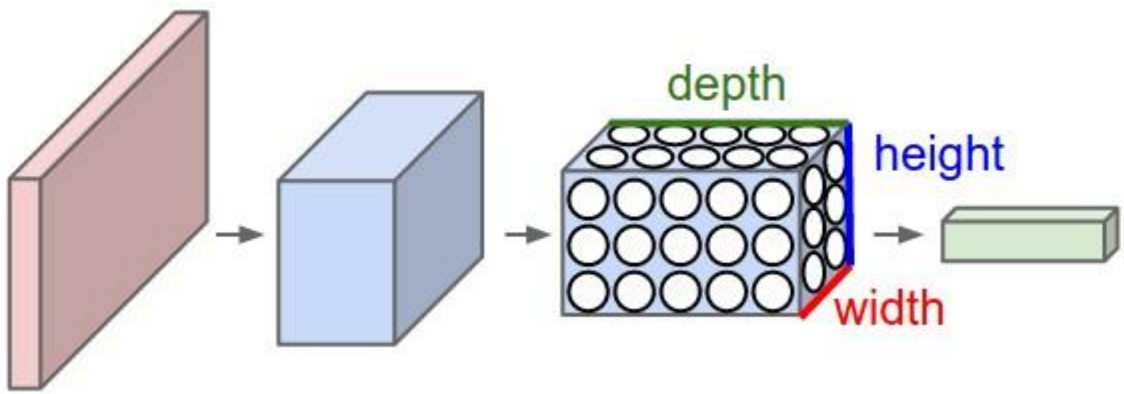


Convolutional Neural Networks

Regular neural network (fully connected):



Convolutional neural network:

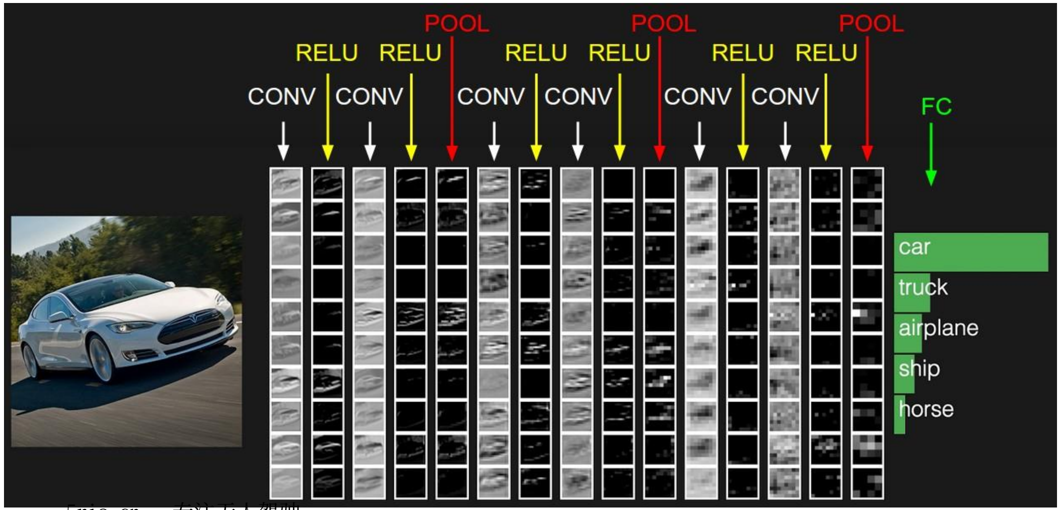


Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

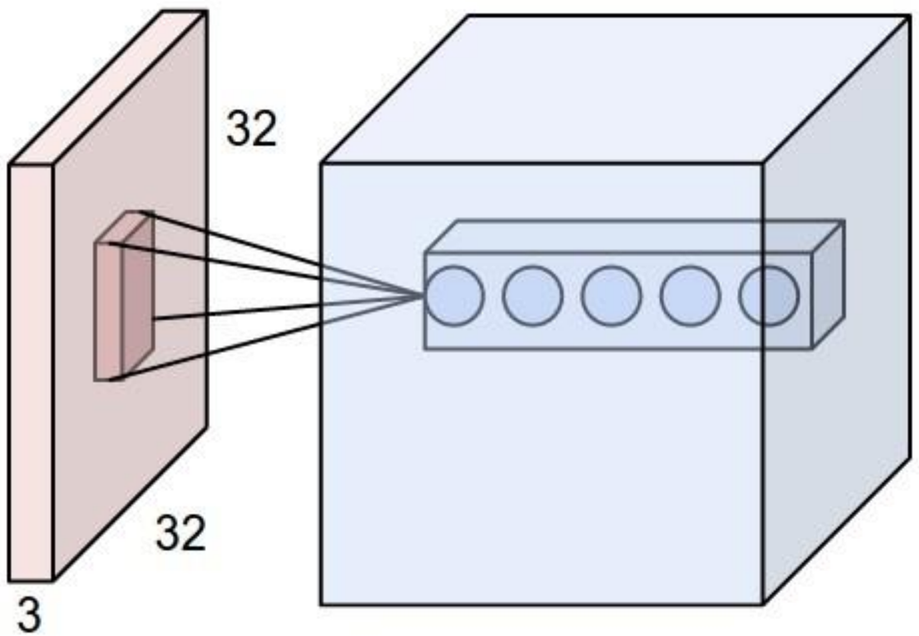
Convolutional Neural Networks: Layers

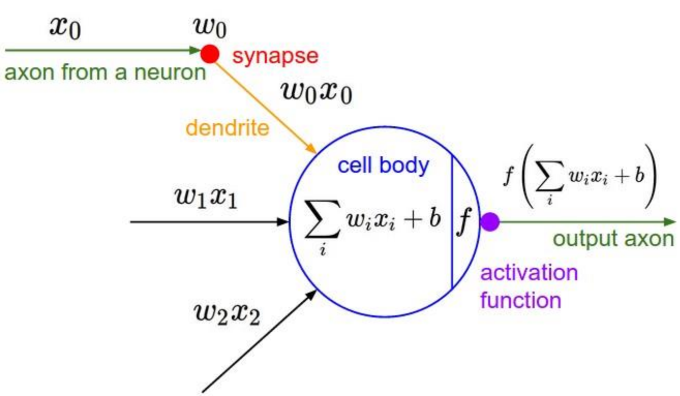
- **INPUT** [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- **RELU** layer will apply an elementwise activation function, such as the *max(0,x)* thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- **FC** (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Layers **highlighted in blue** have learnable parameters.



Dealing with Images: Local Connectivity



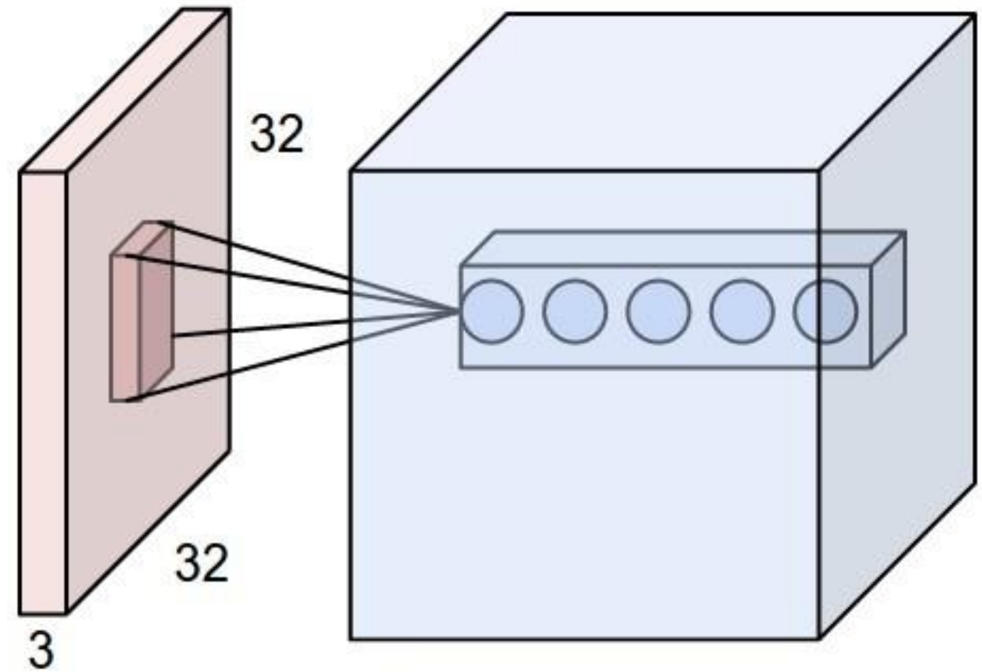


Same neuron. Just more focused (narrow “receptive field”).

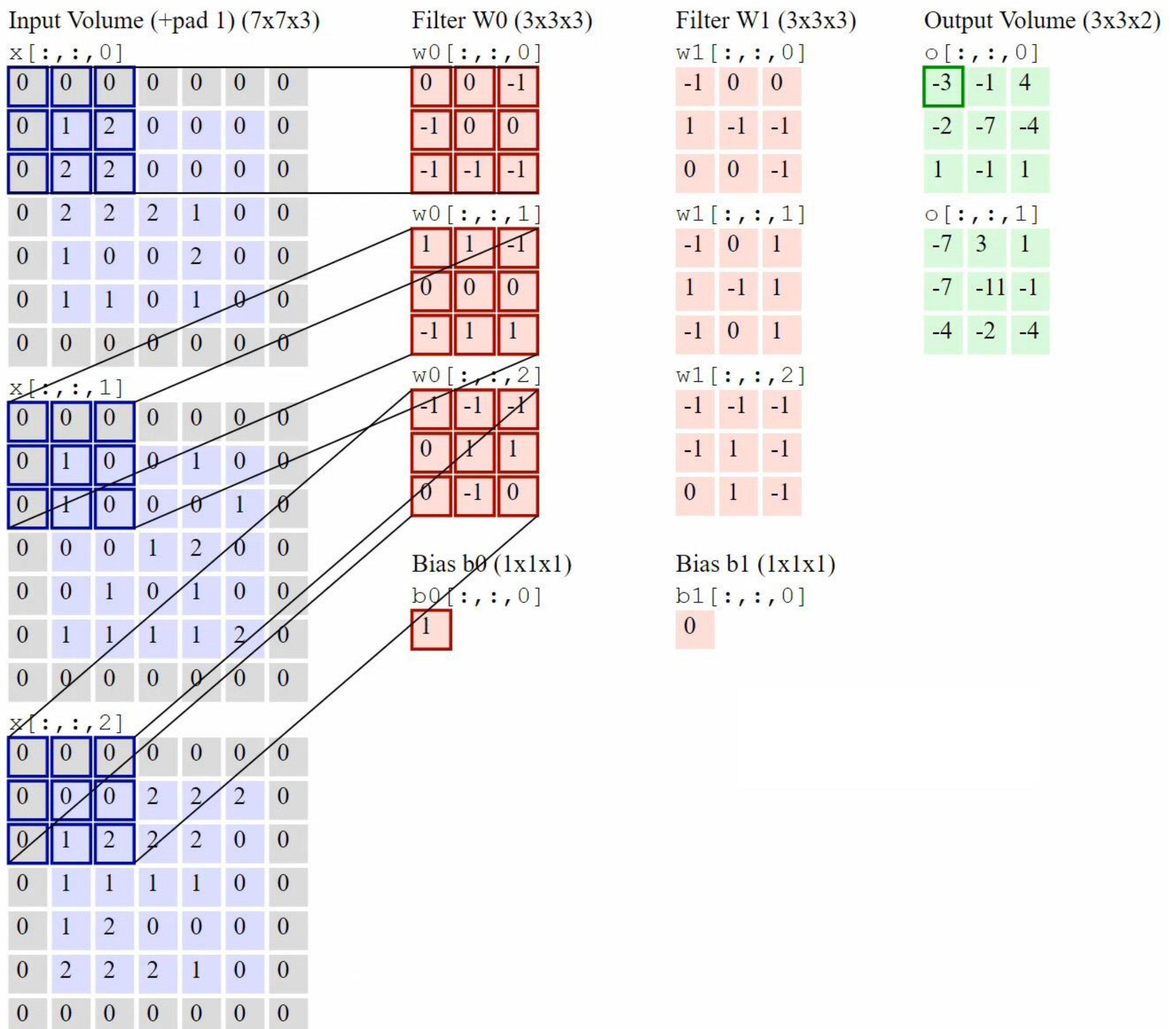
The parameters on a each filter are spatially “shared”

(if a feature is useful in one place, it’s useful elsewhere)

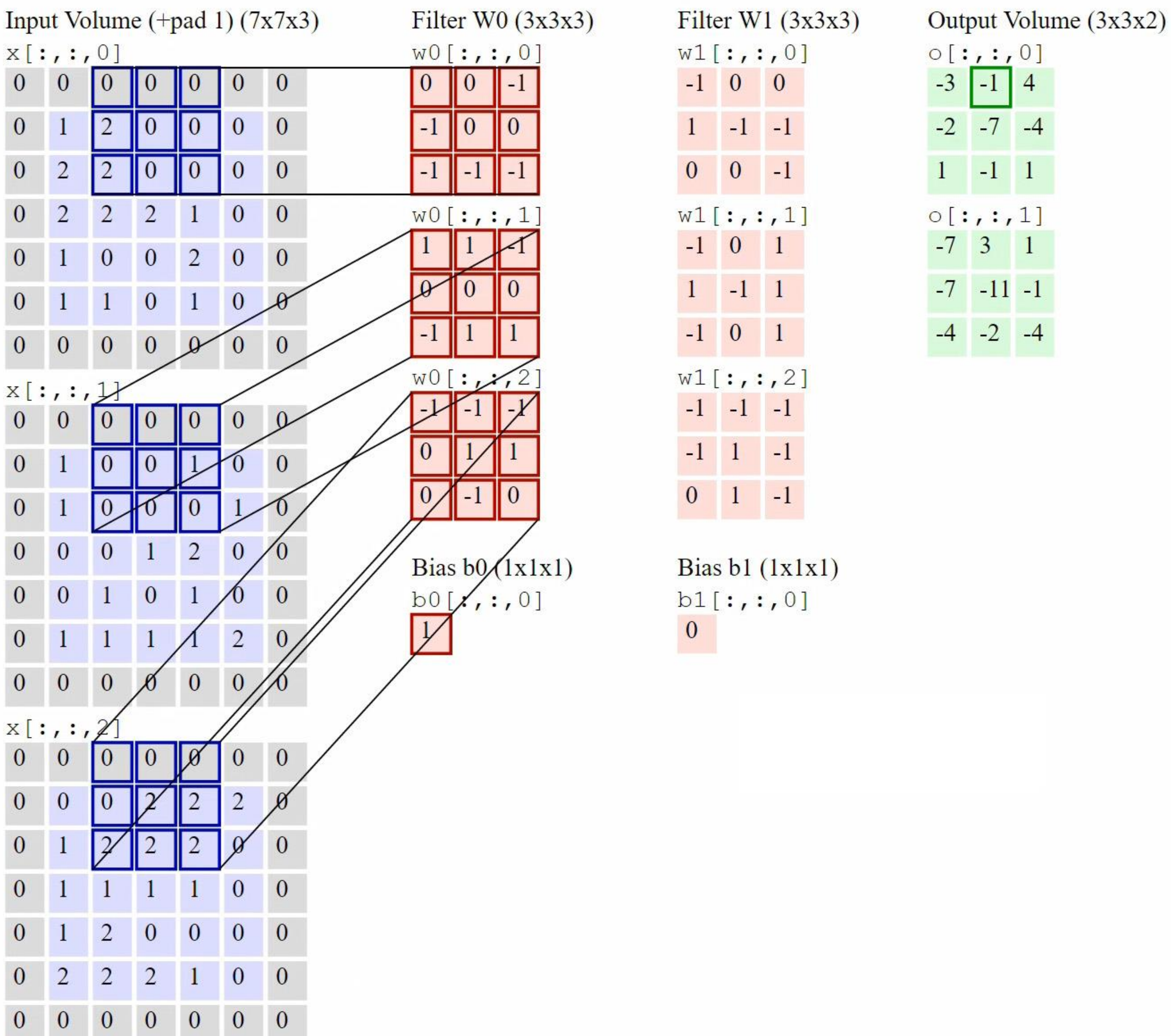
ConvNets: Spatial Arrangement of Output Volume

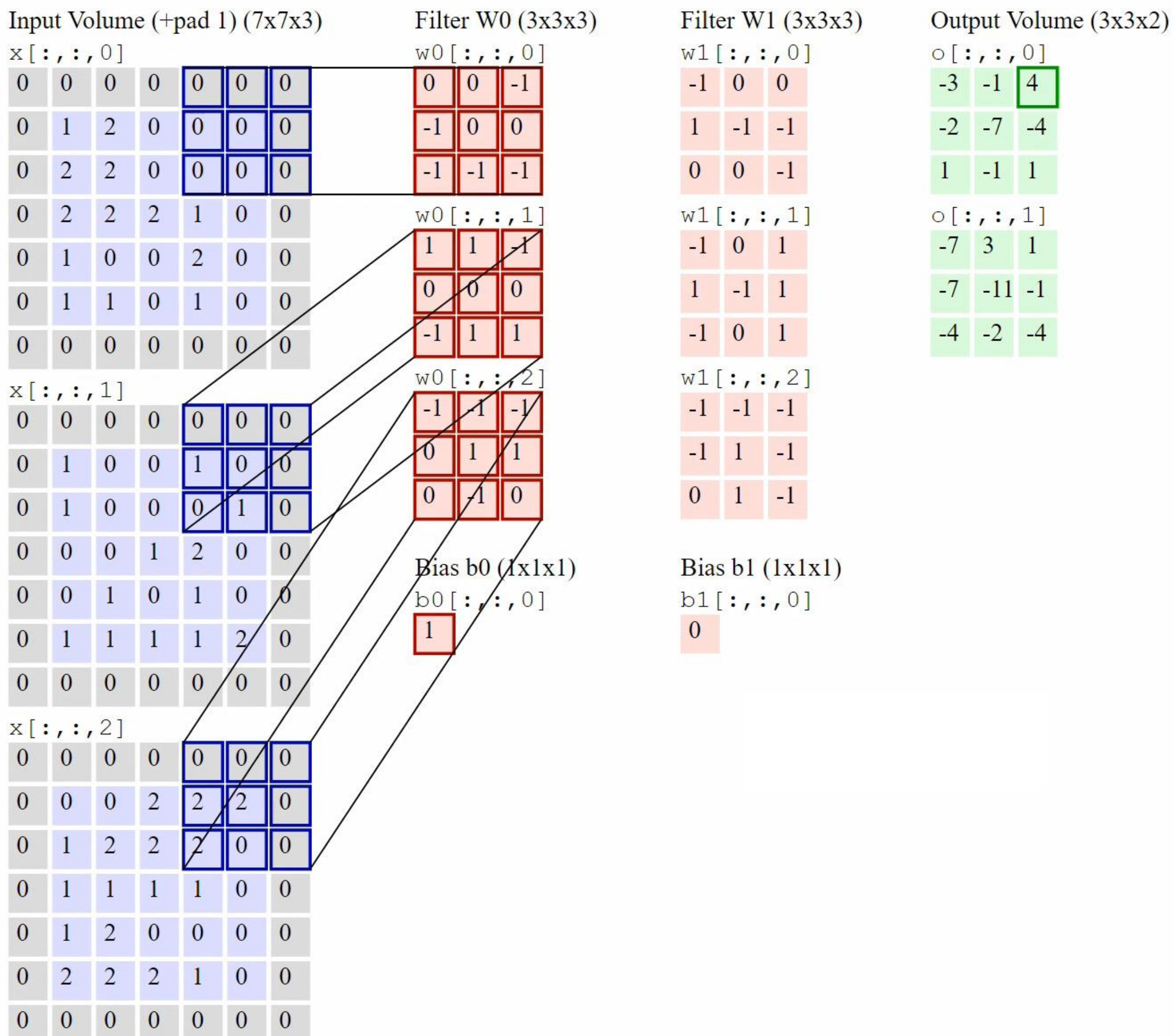


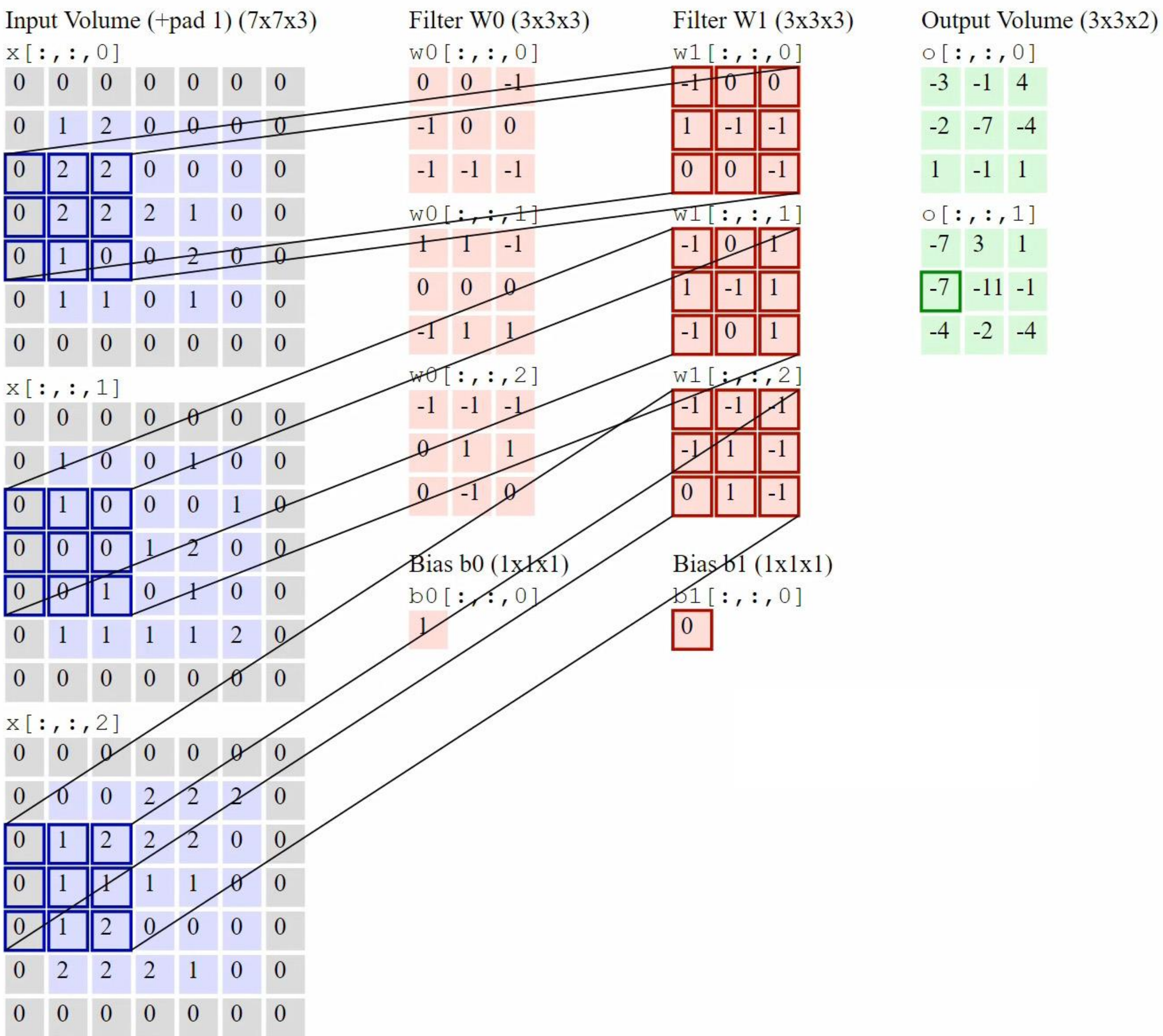
- **Depth:** number of filters
- **Stride:** filter step size (when we “slide” it)
- **Padding:** zero-pad the input



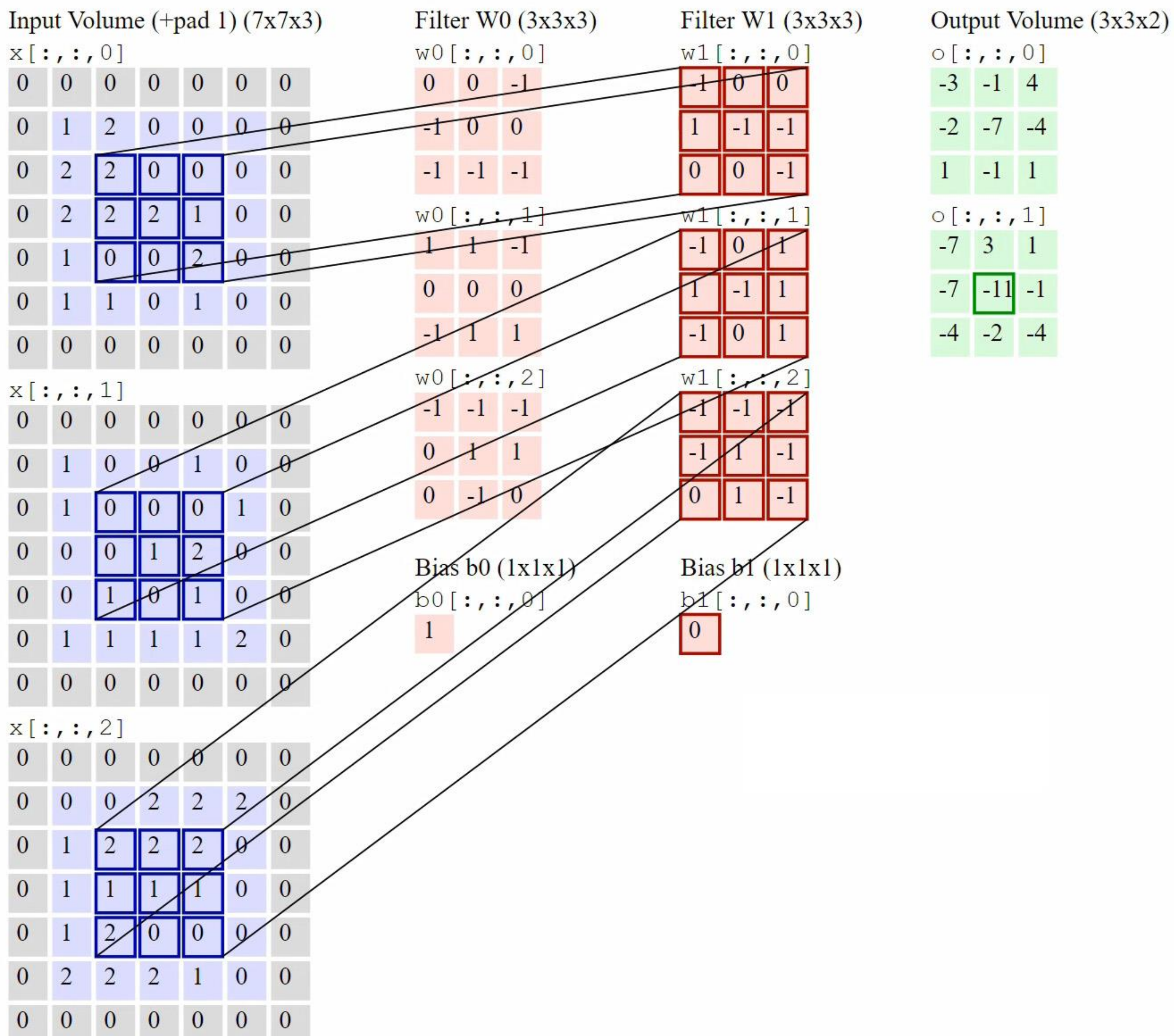
References: [95]

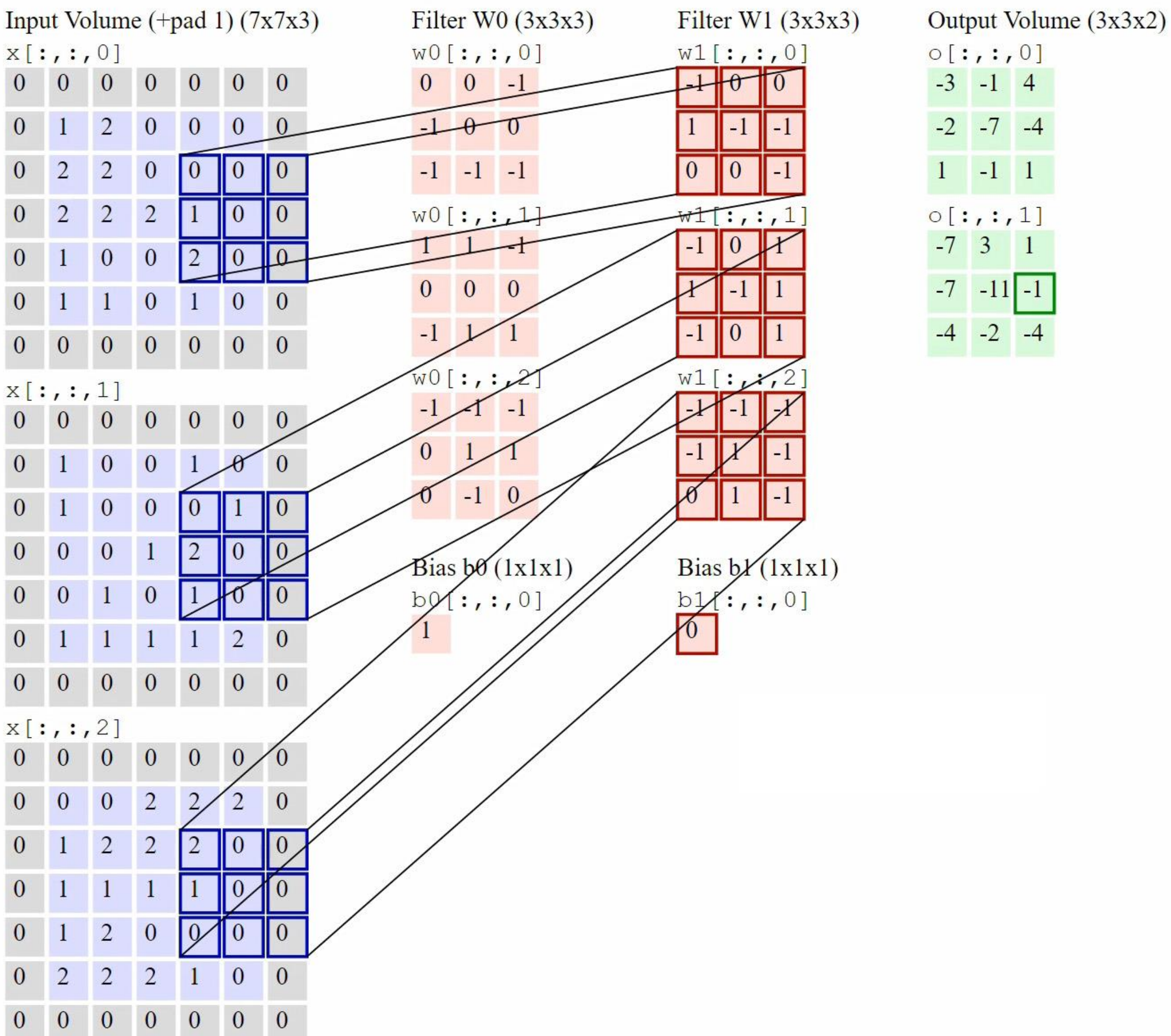

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

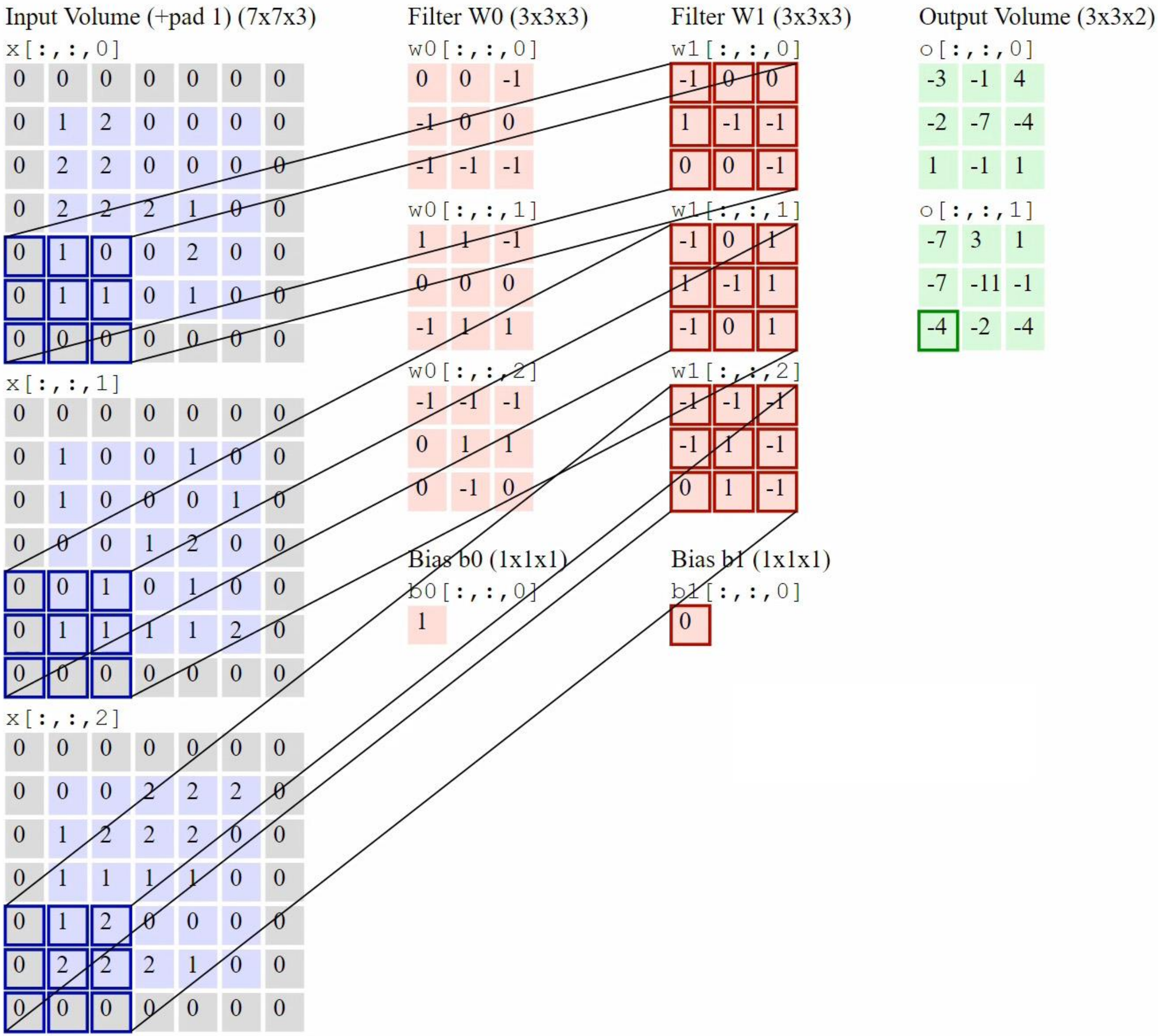

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

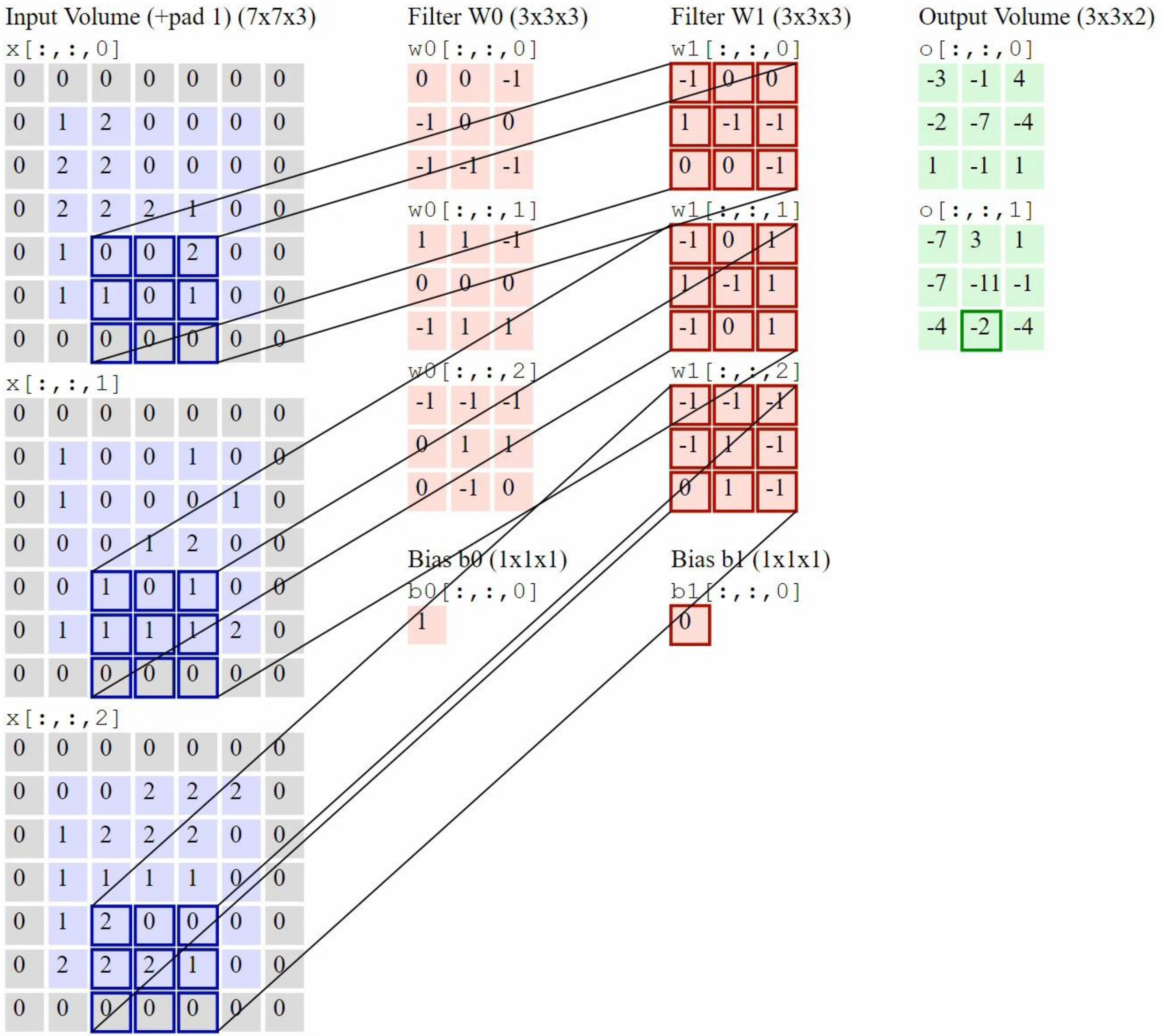


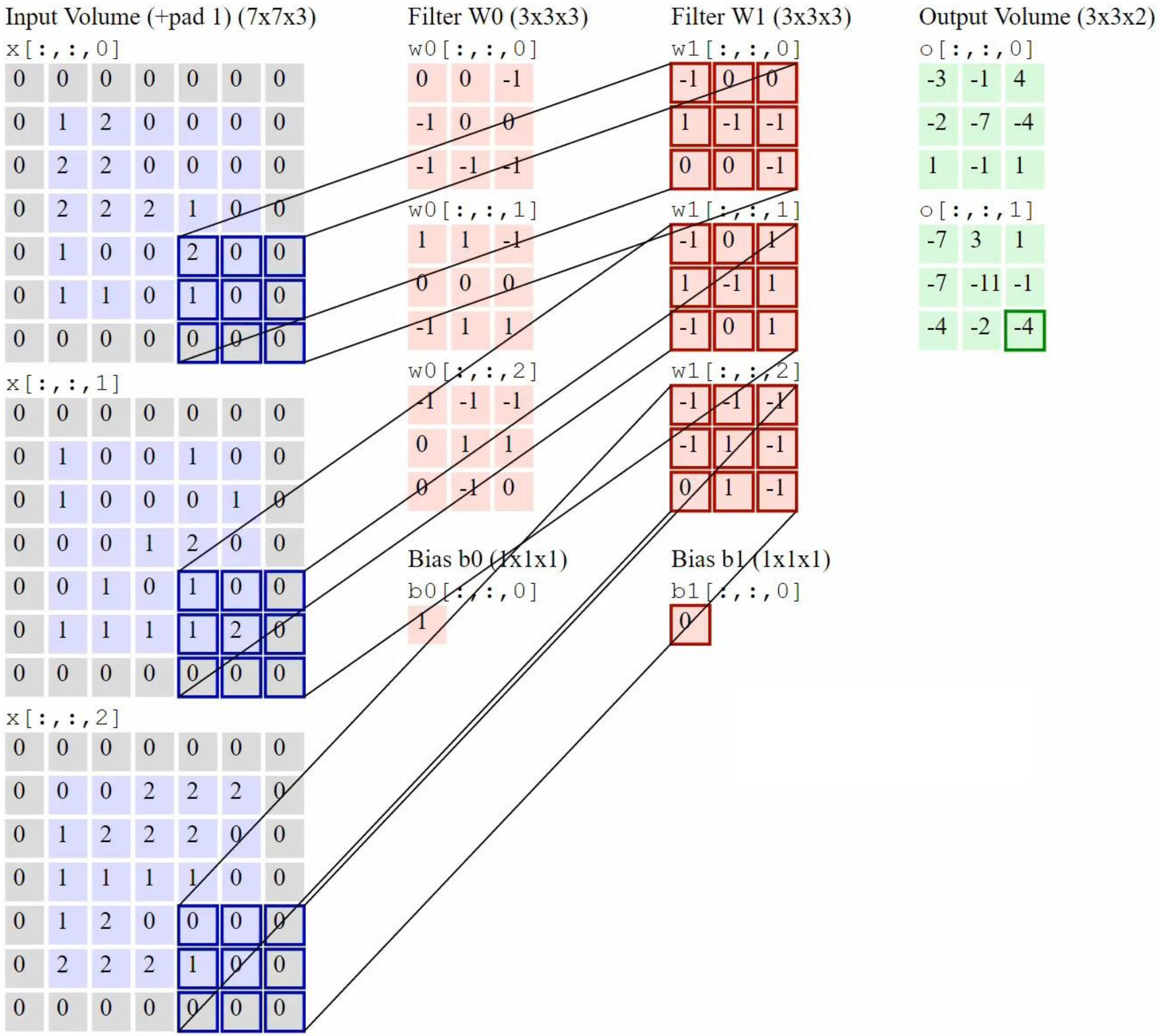
References: [95]


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
x[:,:,1]
x[:,:,2]
Filter W0 (3x3x3)
w0[:,:,0]
w0[:,:,1]
w0[:,:,2]
Filter W1 (3x3x3)
w1[:,:,0]
w1[:,:,1]
w1[:,:,2]
Output Volume (3x3x2)
o[:,:,0]
o[:,:,1]
Bias b0 (1x1x1)
b0[:,:,0]
Bias b1 (1x1x1)
b1[:,:,0]


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 2 0 0 0 0
0 2 2 0 0 0 0
0 2 2 2 1 0 0
0 1 0 0 2 0 0
0 1 1 0 1 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 0 0 1 0 0
0 1 0 0 0 1 0
0 0 0 1 2 0 0
0 0 1 0 1 0 0
0 1 1 1 1 2 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 0 0 2 2 2 0
0 1 2 2 2 0 0
0 1 1 1 1 0 0
0 1 2 0 0 0 0
0 2 2 2 1 0 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
0 0 -1
-1 0 0
-1 -1 -1
w0[:,:,1]
1 1 -1
0 0 0
-1 1 1
w0[:,:,2]
-1 -1 -1
0 1 1
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
-1 0 0
1 -1 -1
0 0 -1
w1[:,:,1]
-1 0 1
1 -1 1
-1 0 1
w1[:,:,2]
-1 -1 -1
-1 1 -1
0 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
-3 -1 4
-2 -7 -4
1 -1 1
o[:,:,1]
-7 3 1
-7 -11 -1
-4 -2 -4

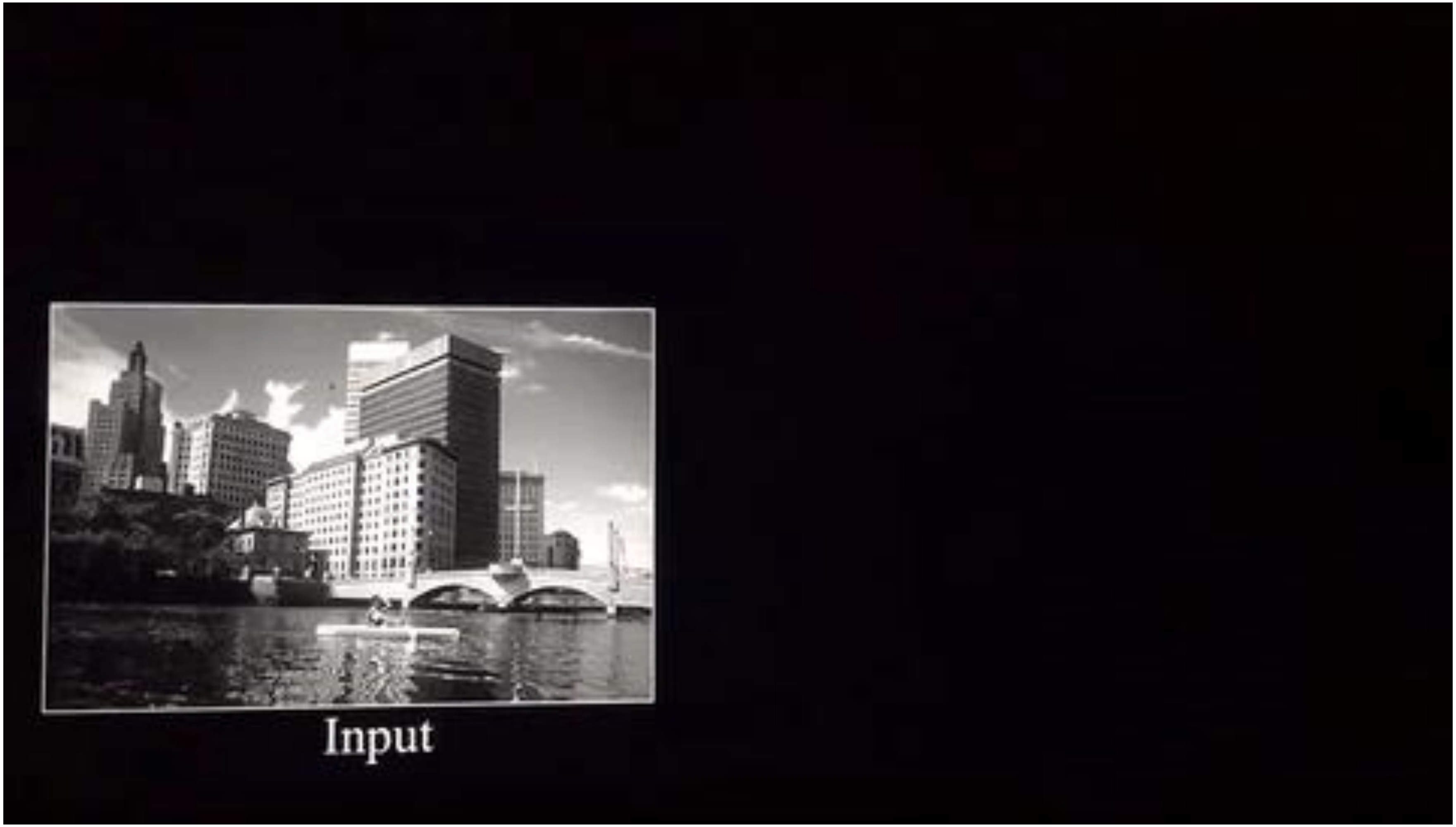


References: [95]

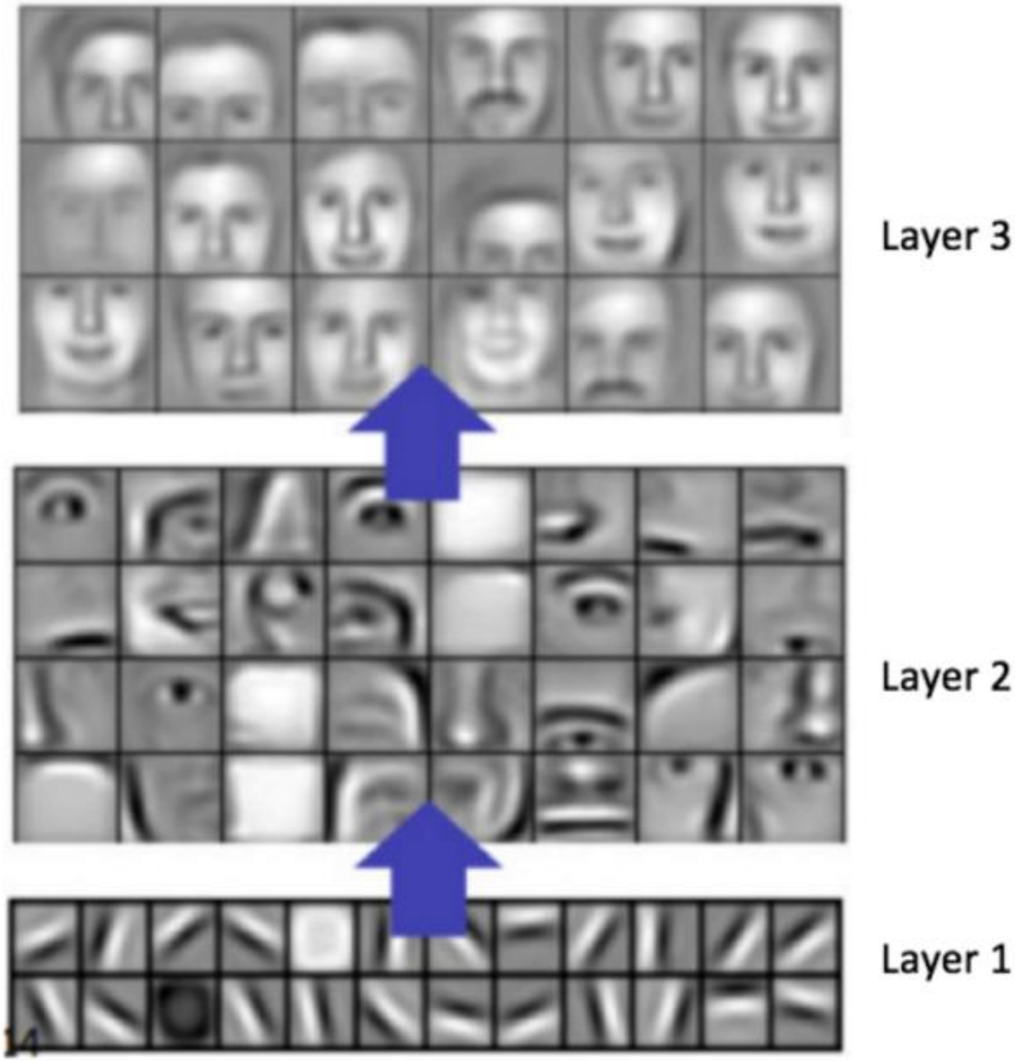
Convolution

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

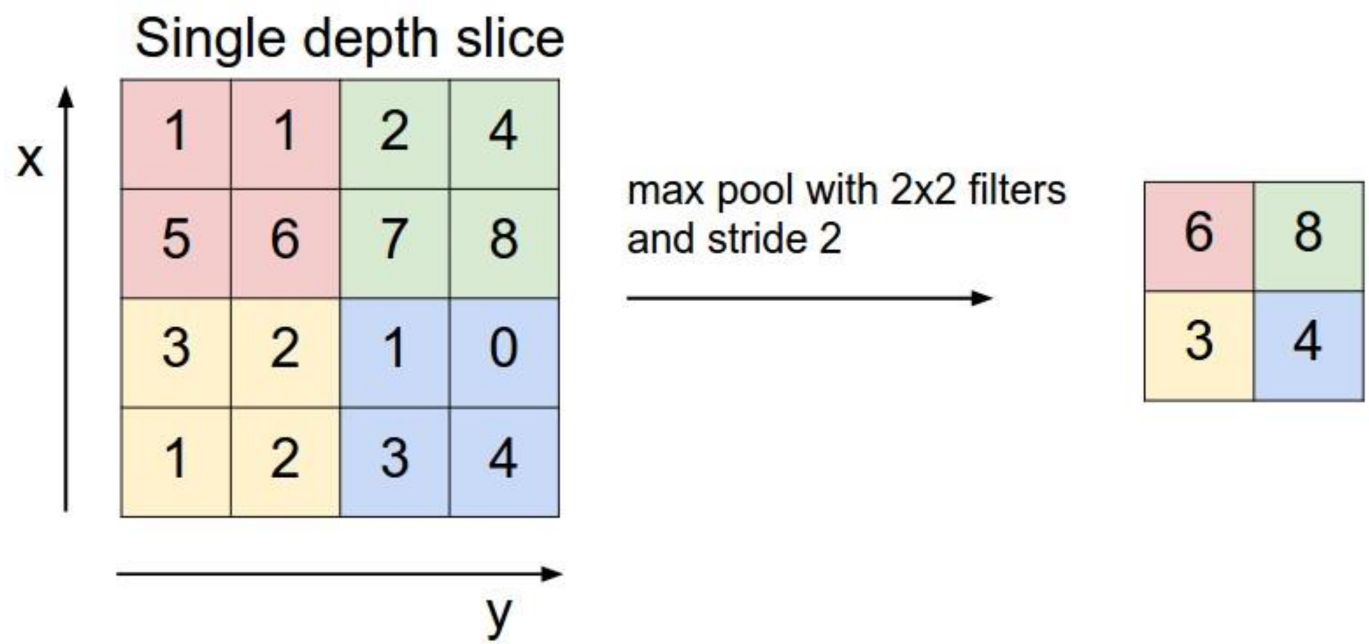
Convolution

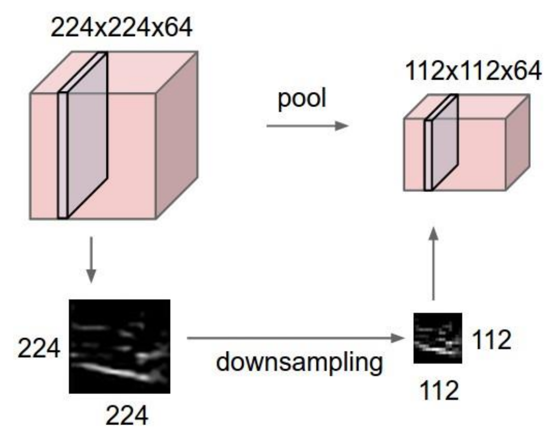


Convolution: Representation Learning

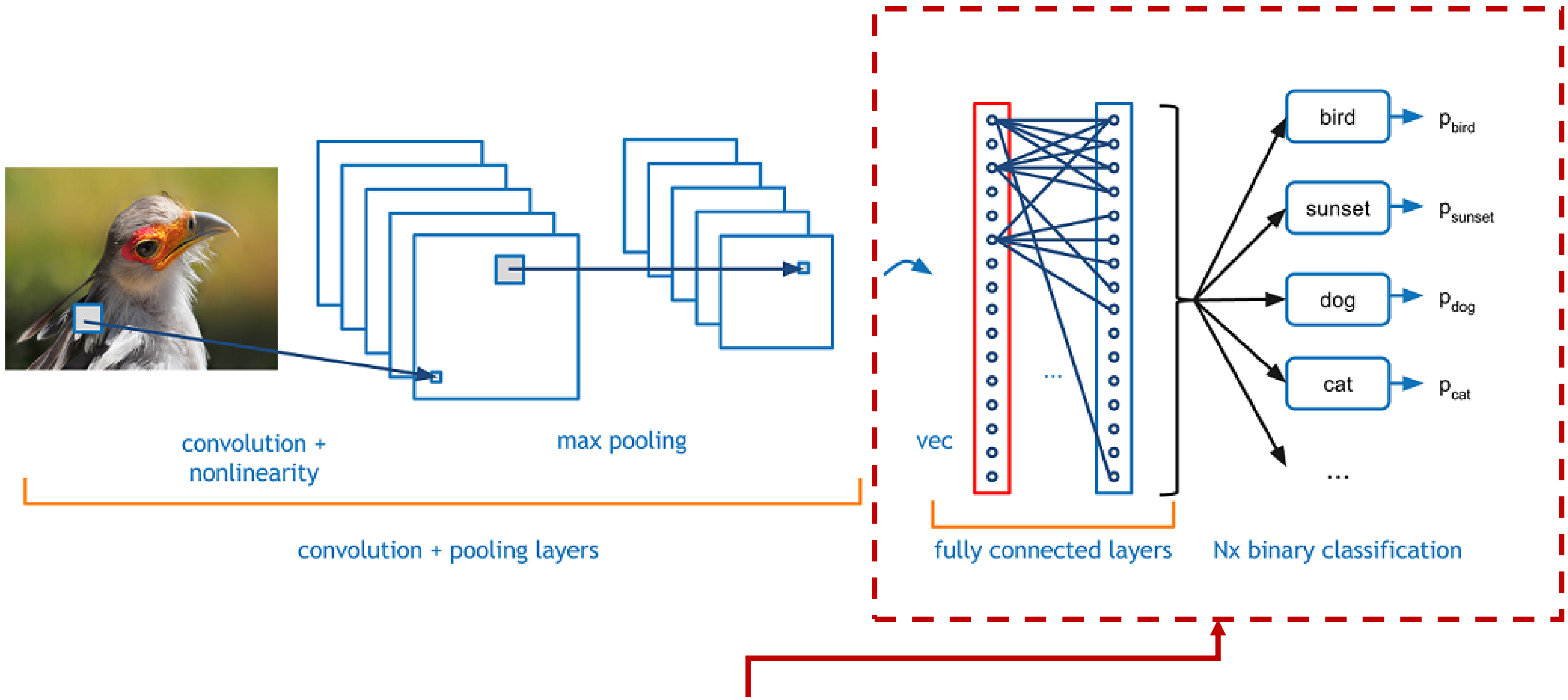


ConvNets: Pooling





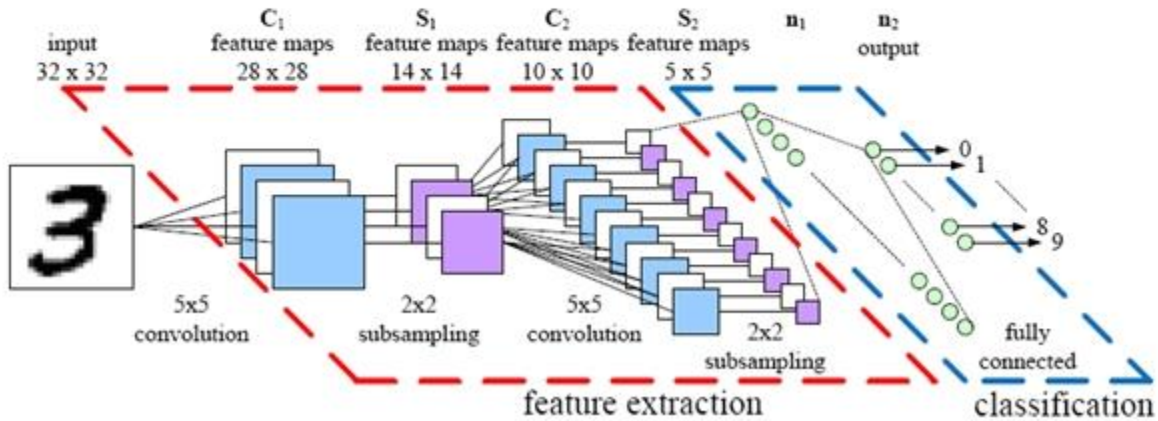
Same Architecture, Many Applications



This part might look different for:

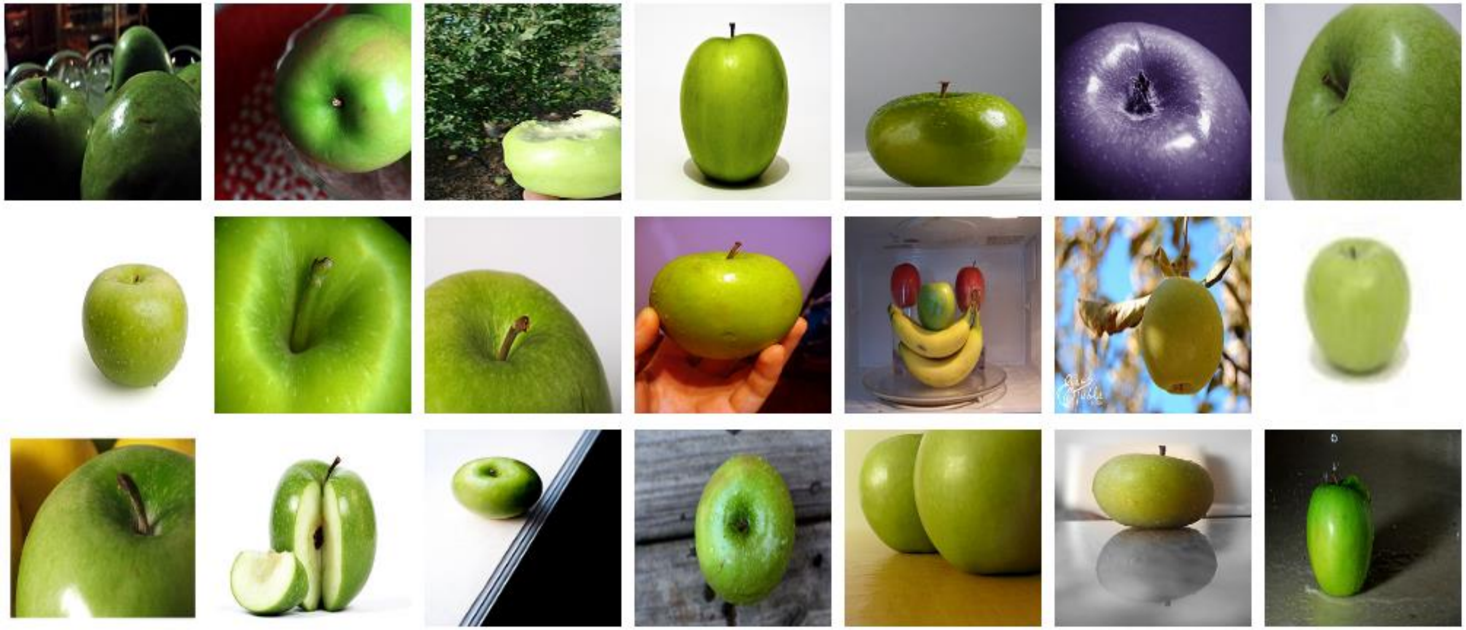
- Different image classification domains
- Image captioning with recurrent neural networks
- Image object localization with bounding box
- Image segmentation with fully convolutional networks
- Image segmentation with deconvolution layers

Object Recognition
Case Study: **ImageNet**



What is ImageNet?

- **ImageNet:** dataset of 14+ million images (21,841 categories)
- Let's take the high level category of **fruit** as an example:
  - Total 188,000 images of fruit
  - There are 1206 Granny Smith apples:



What is ImageNet?

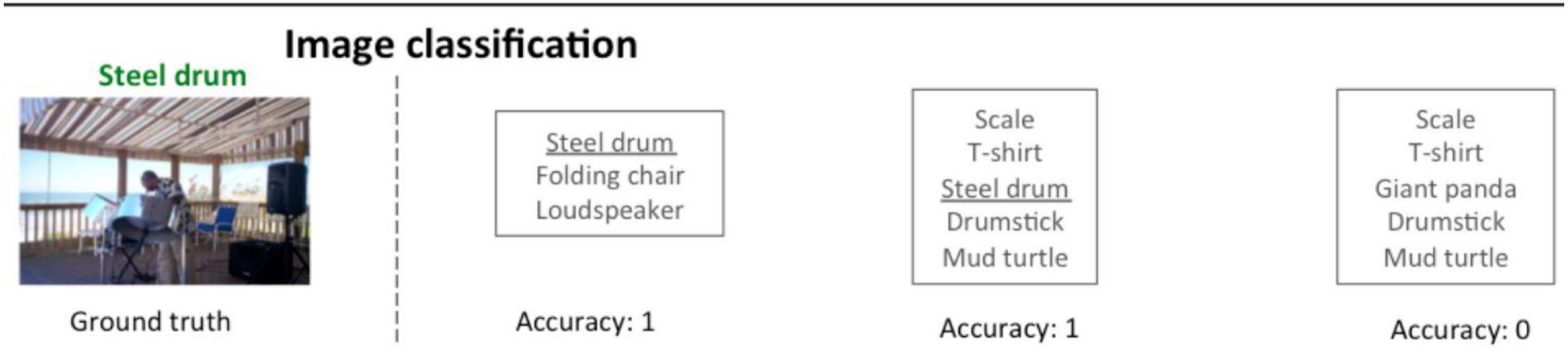
Dataset →
- **ImageNet:** dataset of 14+ million images

Competition →
- **ILSVRC:** ImageNet Large Scale Visual Recognition Challenge

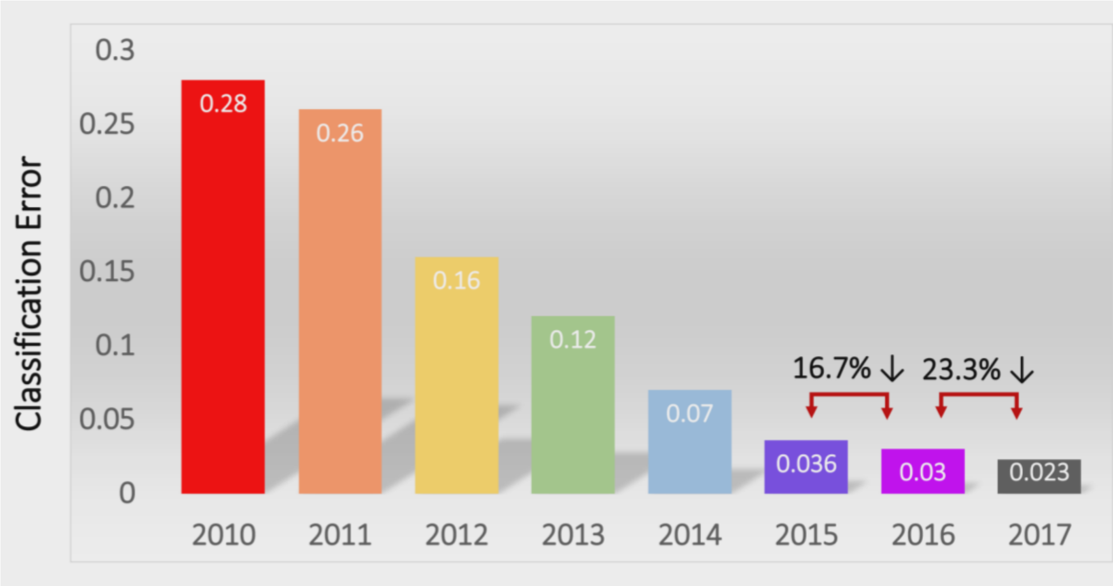
Networks →
- AlexNet (2012)
- ZFNet (2013)
- VGGNet (2014)
- GoogLeNet (2014)
- ResNet (2015)
- CUImage (2016)
- SENet (2017)

ILSVRC Challenge Evaluation for Classification

- Top 5 error rate:
  - You get 5 guesses to get the correct label

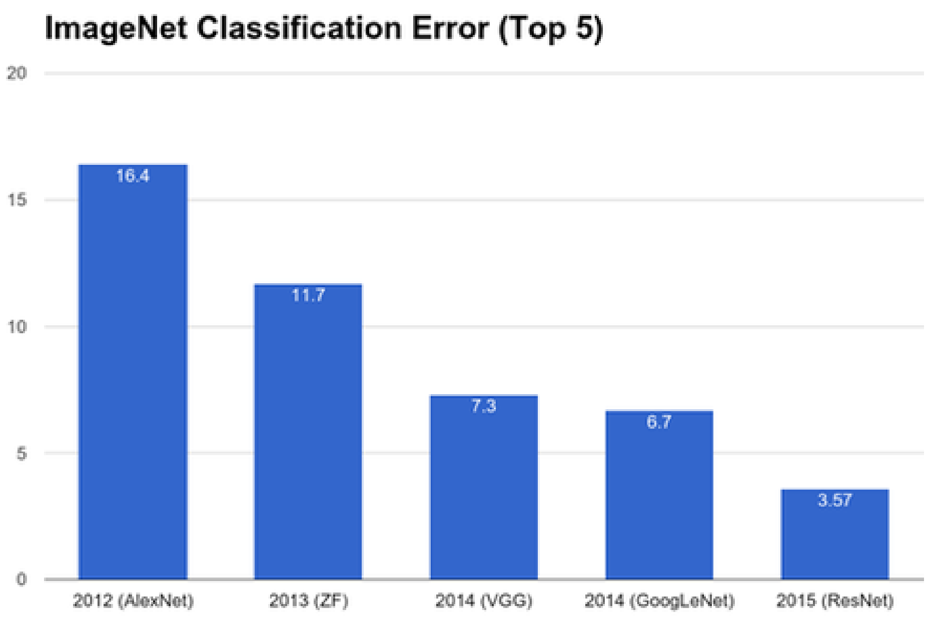


- ~20% reduction in accuracy for Top 1 vs Top 5
- Human annotation is a binary task: “apple” or “not apple”



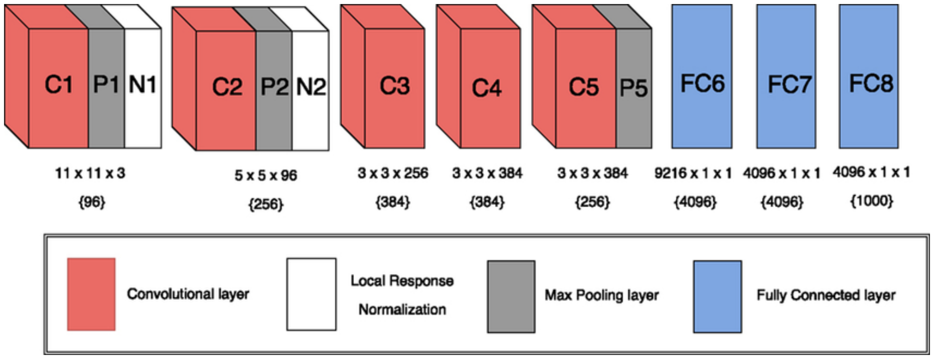
- Human error: 5.1%
  - Surpassed in 2015

- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform: 3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models
- **SENet (2017): 2.99% to 2.251%**
  - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.



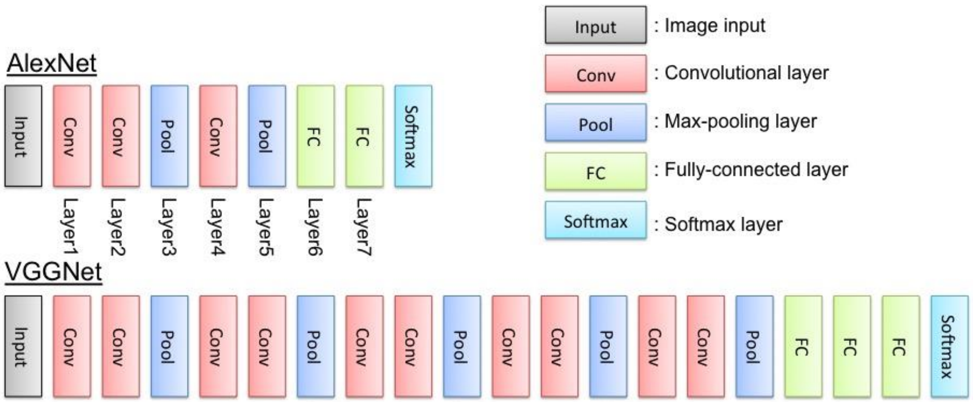
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

References: [90]



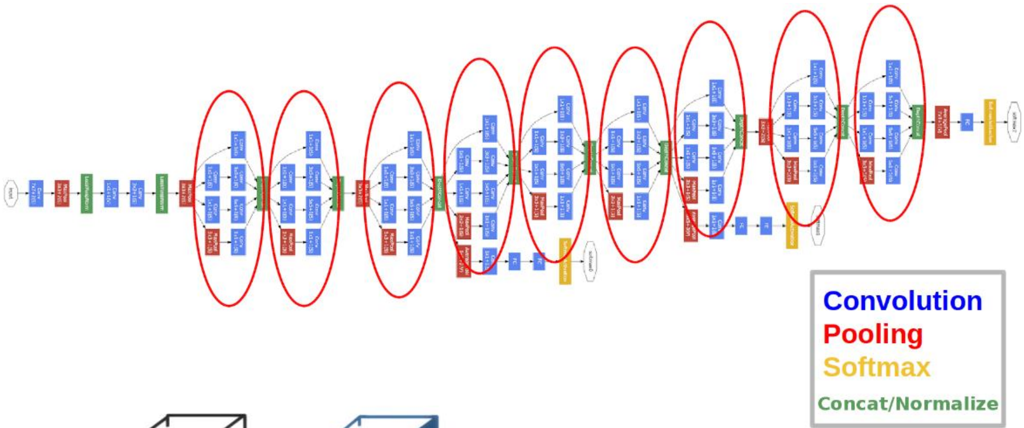
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

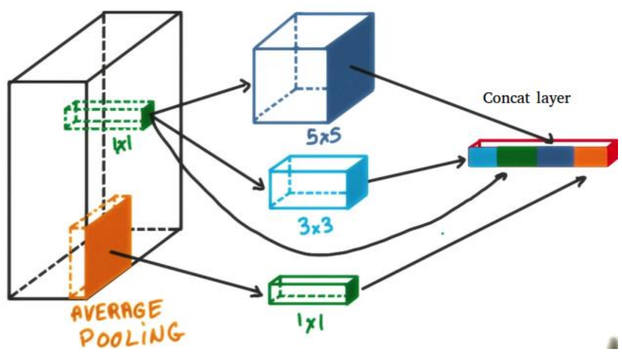
Krizhevsky et al. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

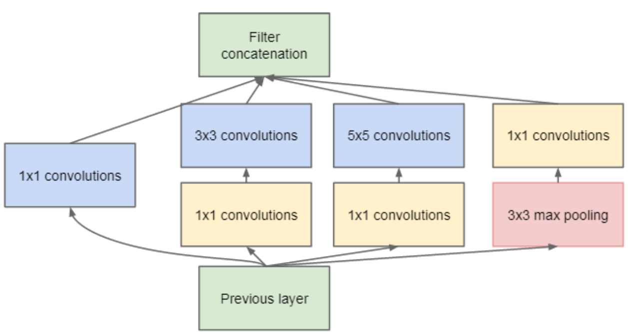


Simonyan et al. "Very deep convolutional networks for large-scale image recognition." 2014.

- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models



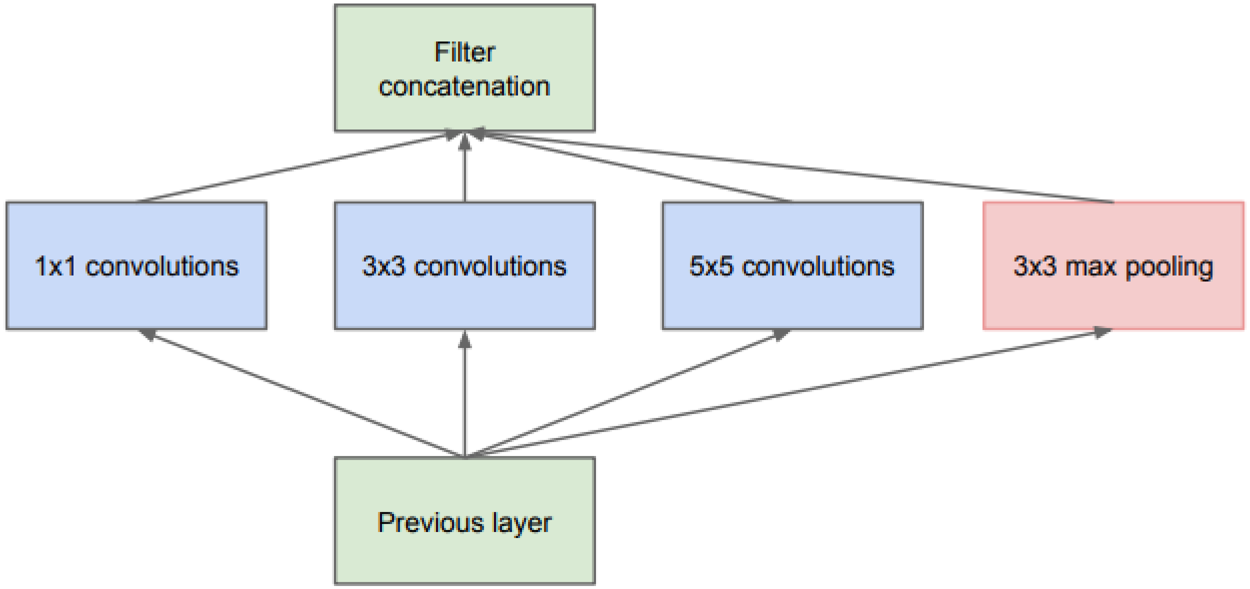




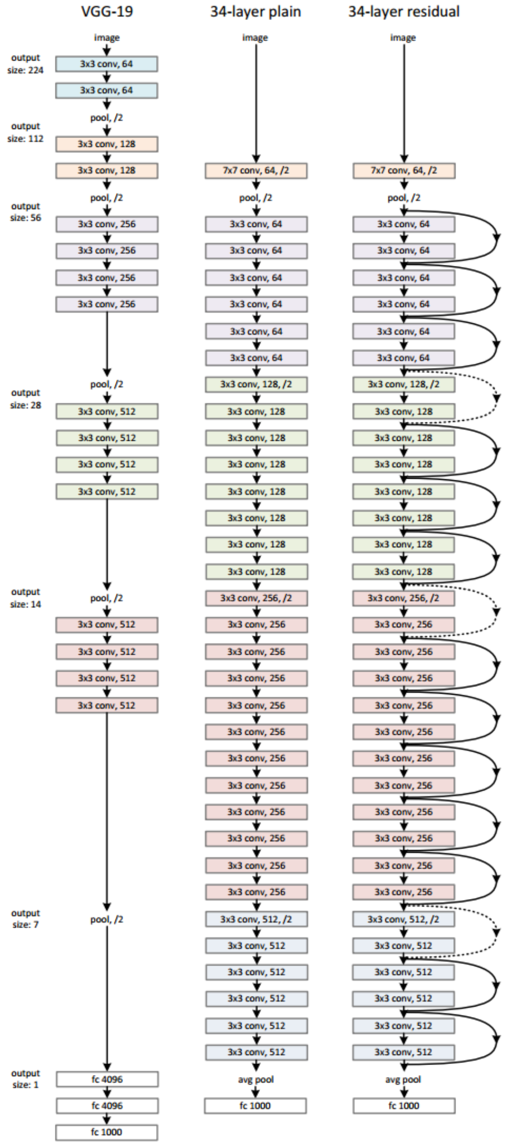
Szegedy et al. "Going deeper with convolutions." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

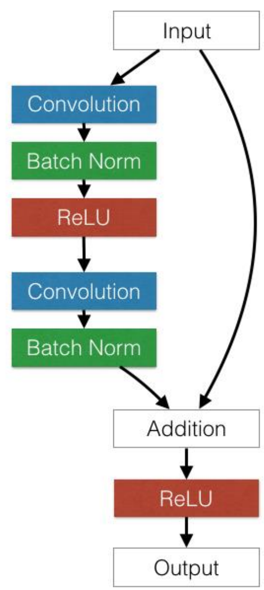
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

Inception Module



- **Process:** do different size convolutions, and concatenate
- Convolution sizes:
  - Smaller convolutions: local features
  - Larger convolutions: high-abstracted features
- **Result**: Fewer parameters and better performance

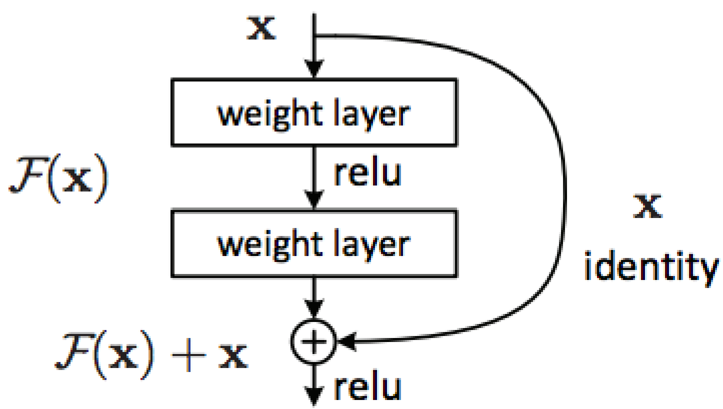




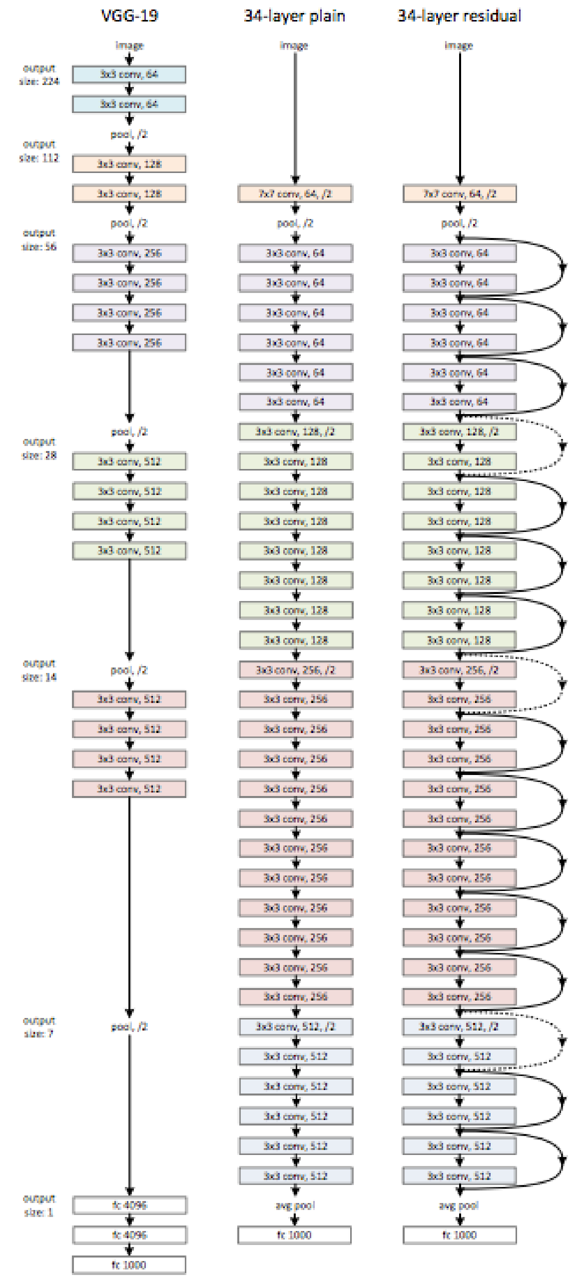
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

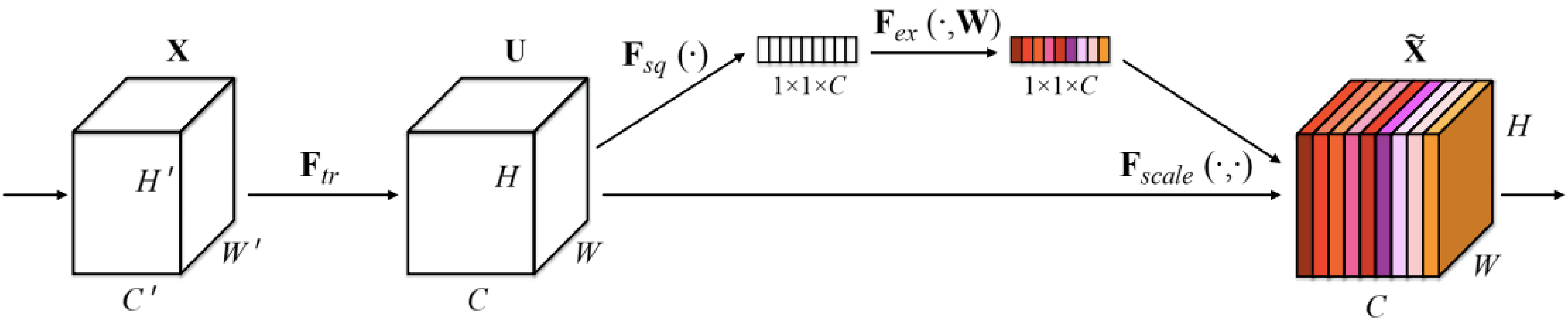
Residual Block



- **Initial Observation:**
  - Network depth often increases representation power, but is harder to train.
- Residual Block:
  - Repeat a simple network block (think: RNN)
  - Pass input along without transformation: help ensure that each layer learns something new

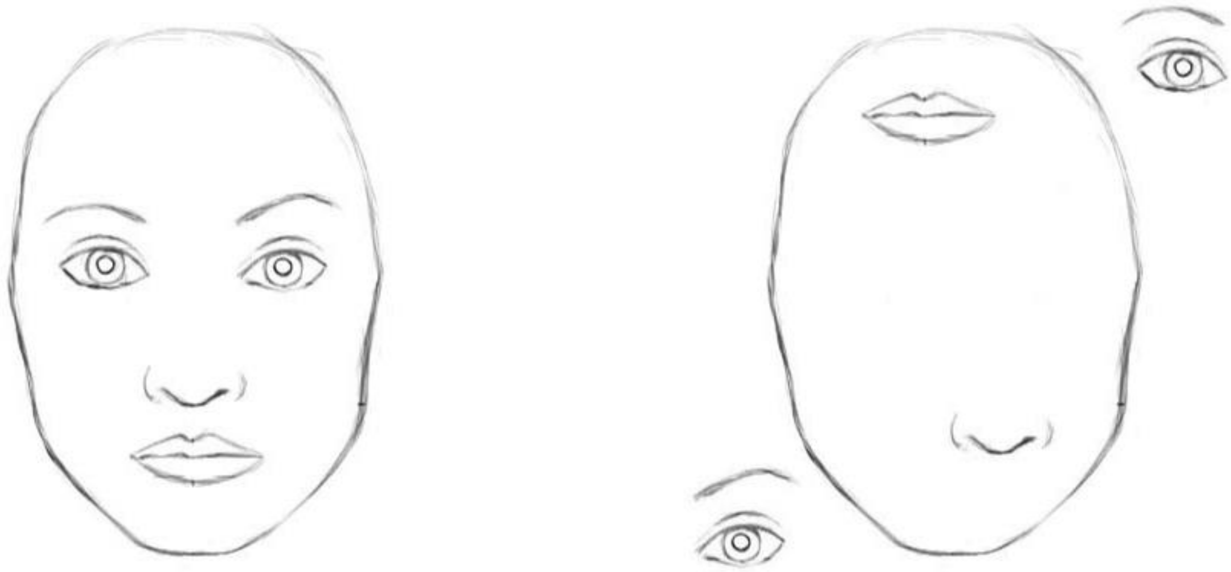


SENet: Squeeze-and-Excitation Networks



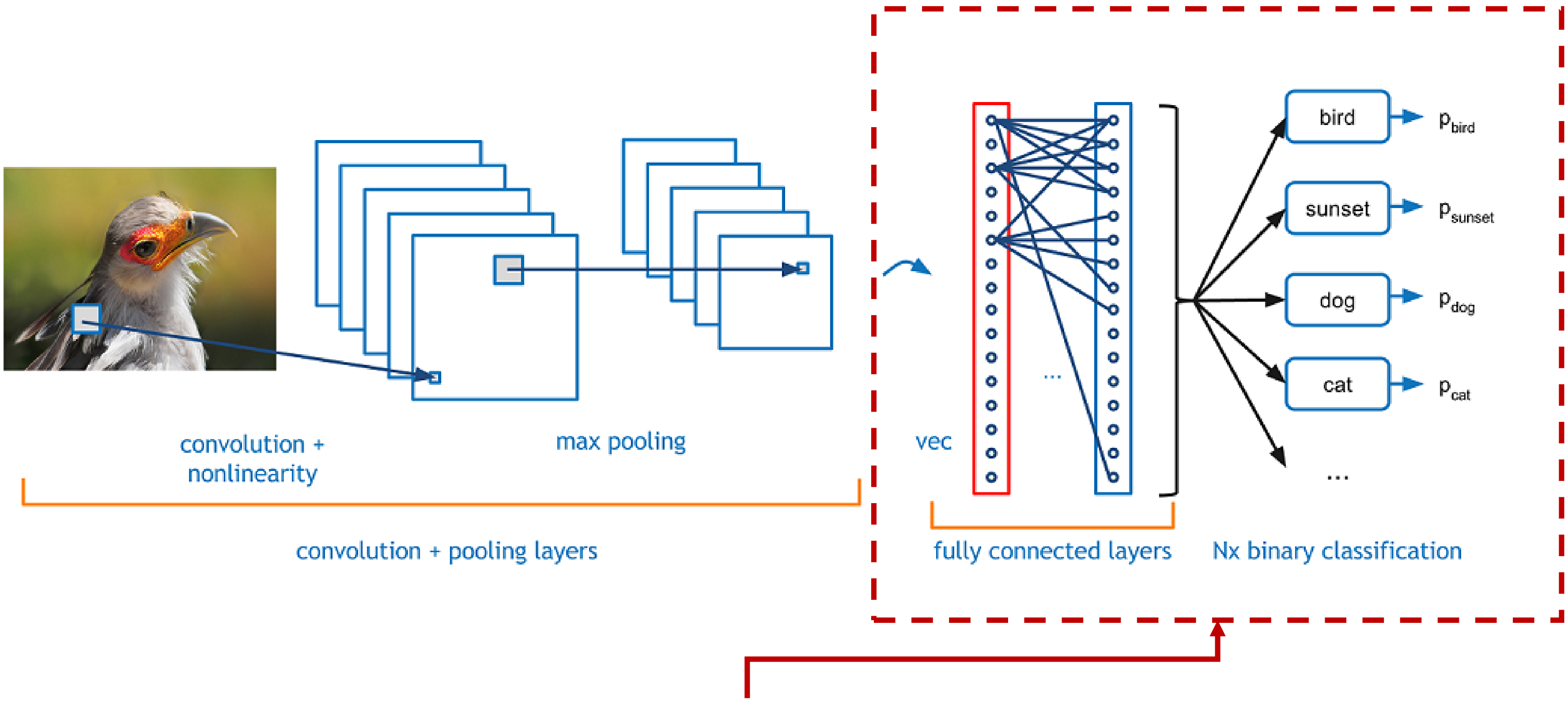
- **Content-aware channel weighting:** Add parameters to each channel of a convolutional block so that the network can adaptively adjust the weighting of each feature map
- This approach is simple and can be added to any model
  - **Takeaway for thought:** Parameterize everything (that's cost-effective) including higher-order hyper-parameters.

Capsule Networks (Hinton)



- A CNN see both images as the same. The problem:
  - *Internal data representation of a convolutional neural network does not take into account important spatial hierarchies between simple and complex objects.*
- See upcoming online-only lecture on capsule networks.

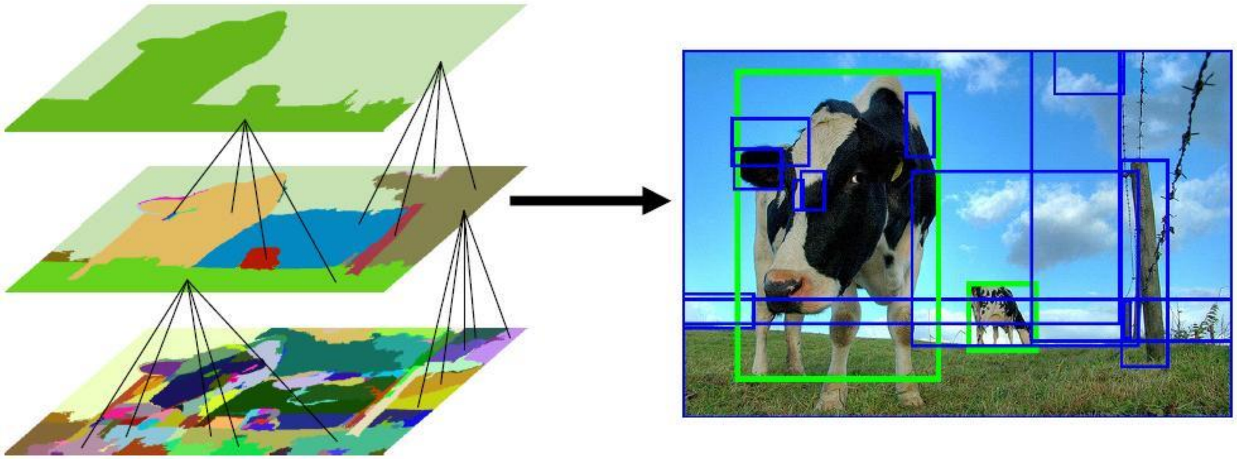
Same Architecture, Many Applications

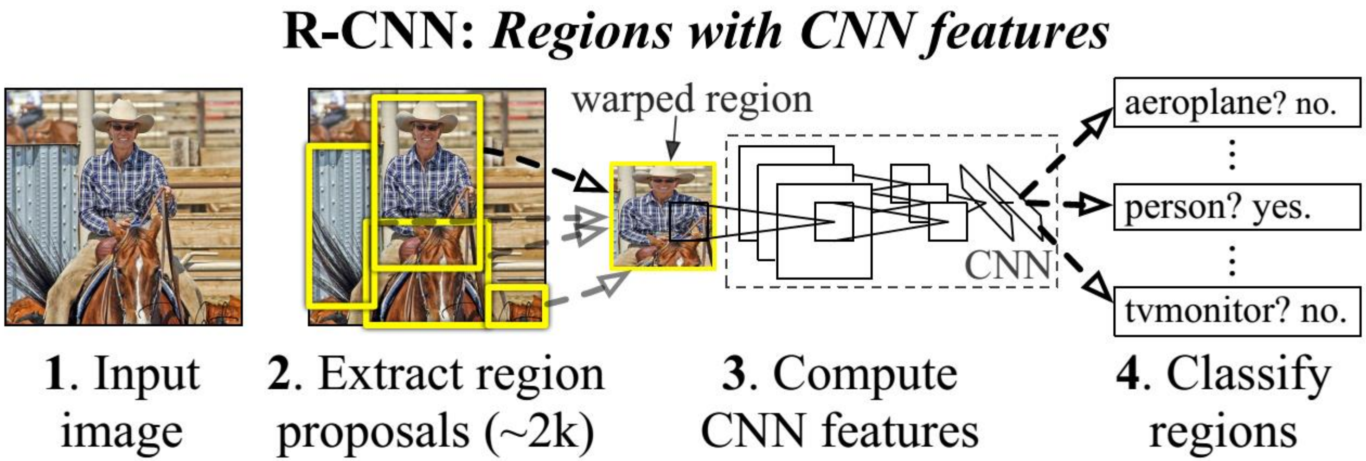


This part might look different for:

- Different image classification domains
- Image captioning with recurrent neural networks
- Image object localization with bounding box
- Image segmentation with fully convolutional networks
- Image segmentation with deconvolution layers

Object Detection





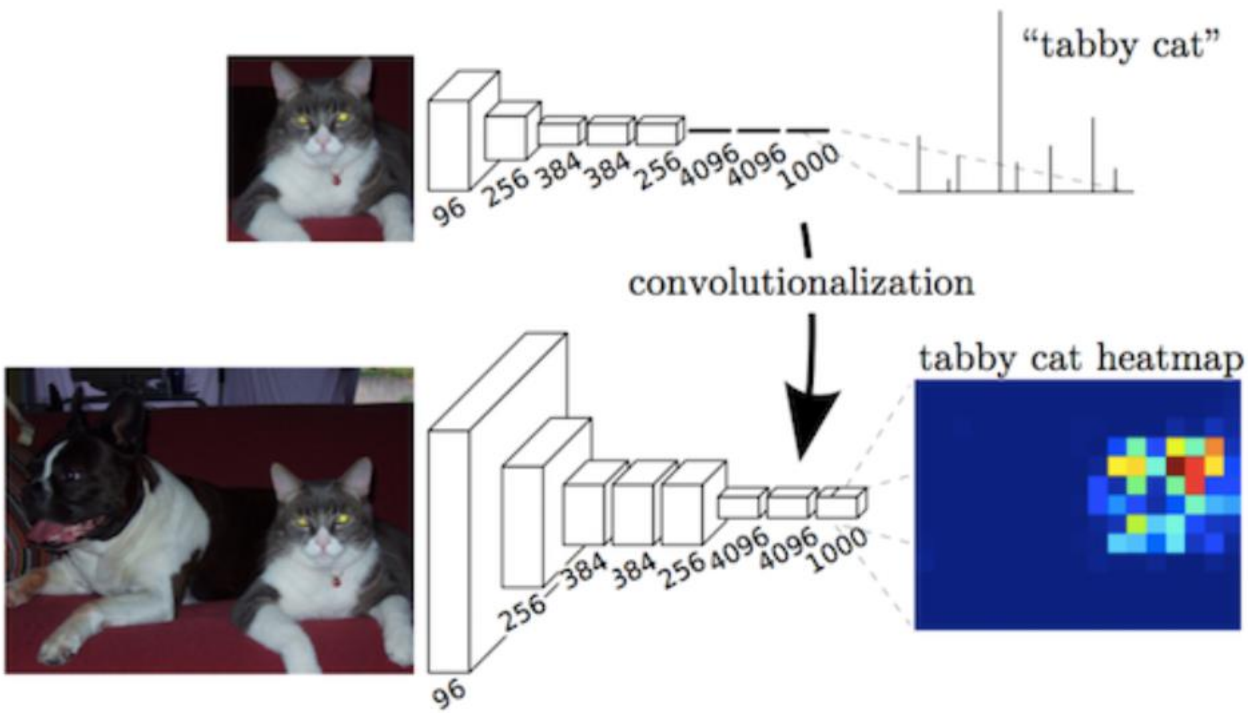
Fully Convolutional Networks

- **Goal:** Classify every pixel in an image.
- **Difficulty:** Hard
- Why?
  - When precise boundaries of objects matter (medical, driving)
  - Useful for fusing with other sensors (LIDAR)

FCN (Nov 2014)

Paper: "Fully Convolutional Networks for Semantic Segmentation"

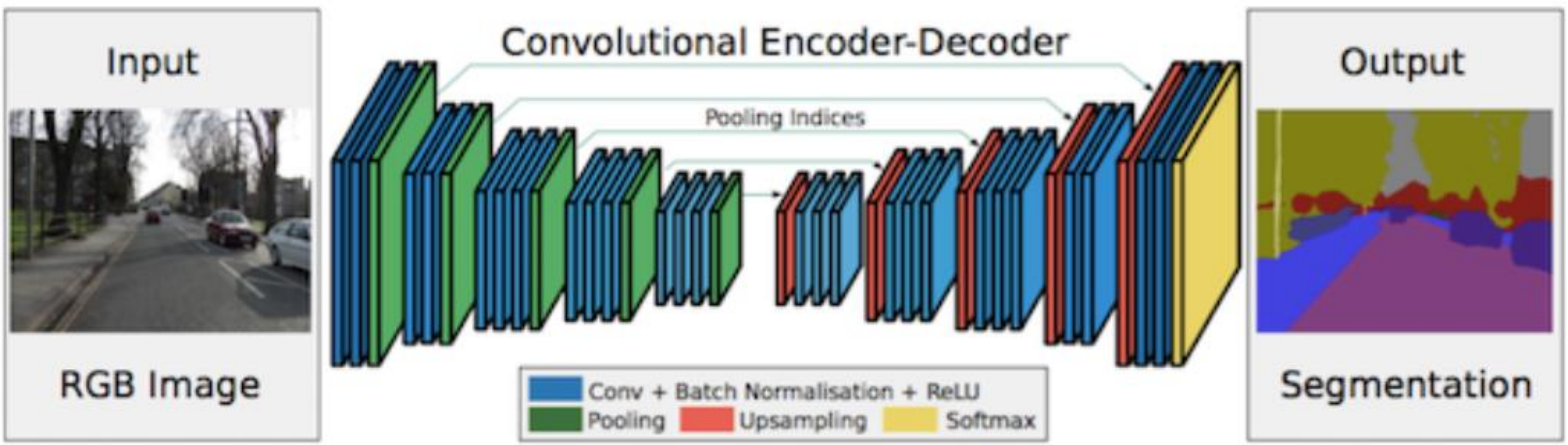
- Repurpose Imagenet pretrained nets
- Upsample using deconvolution
- Skip connections to improve coarseness of upsampling



SegNet (Nov 2015)

Paper: “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”

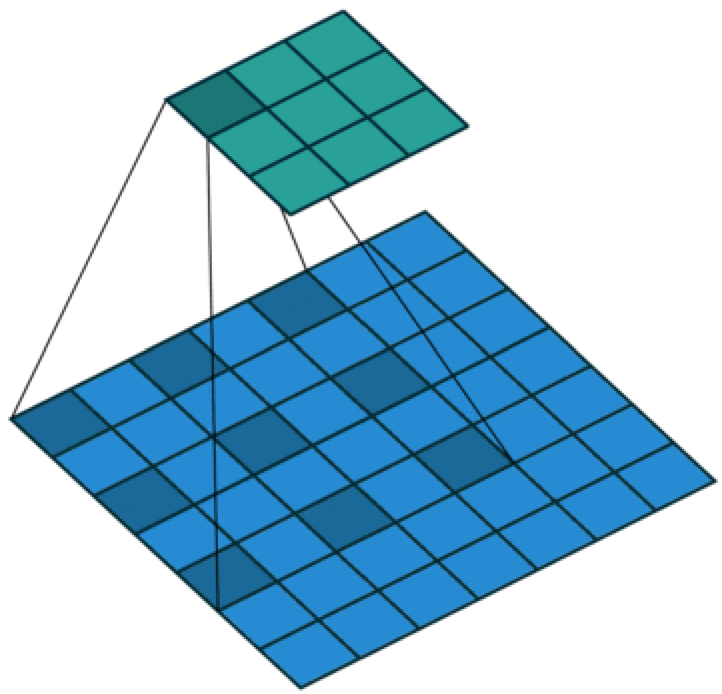
- Maxpooling indices transferred to decoder to improve the segmentation resolution.



Dilated Convolutions (Nov 2015)

Paper: “Multi-Scale Context Aggregation by Dilated Convolutions”

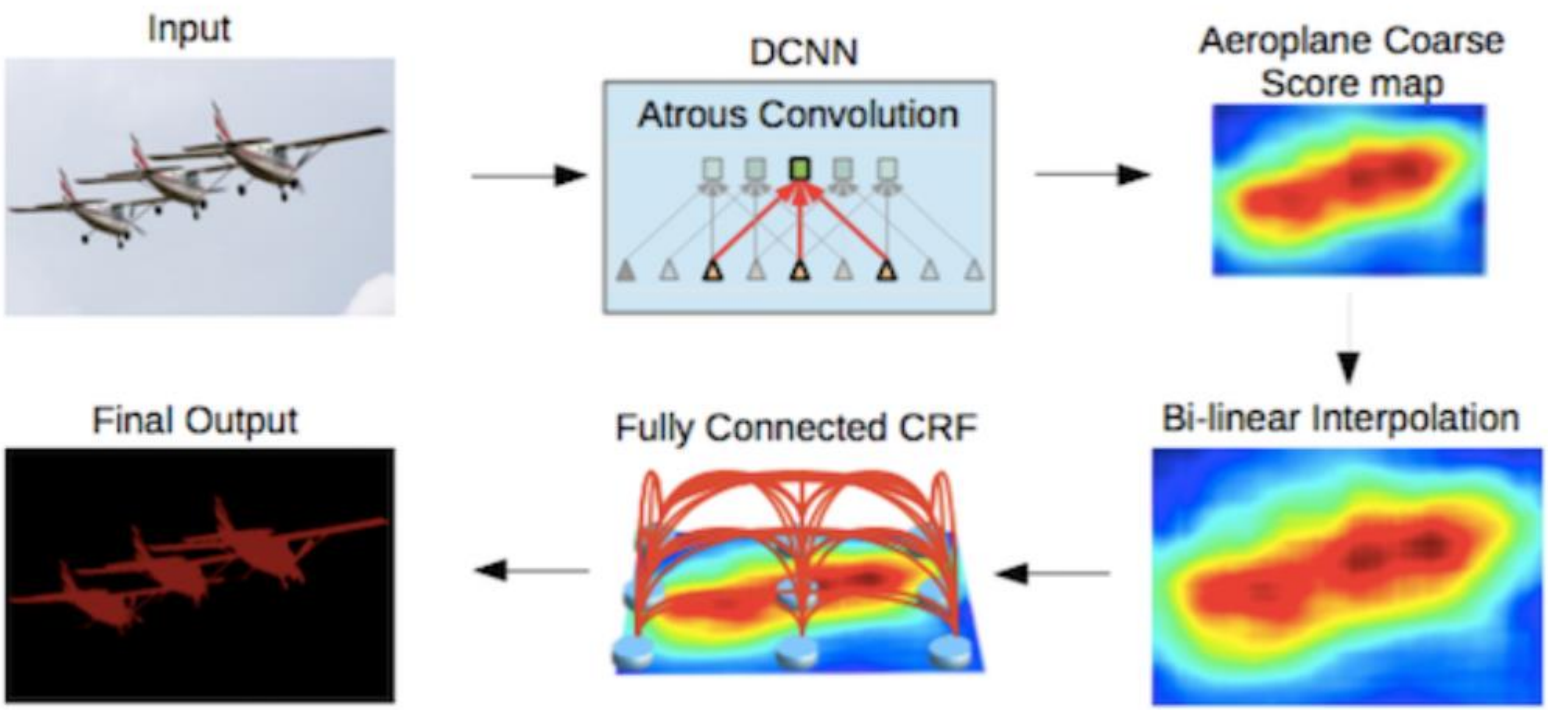
- Since pooling decreases resolution:
  - Added “dilated convolution layer”
- Still interpolate up from 1/8 of original image size



DeepLap v1, v2 (Jun 2016)

Paper: "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs"

- Added fully-connected Conditional Random Fields (CRFs) – as a post-processing step
  - Smooth segmentation based on the underlying image intensities



Key Aspects of Segmentation

- **Fully convolutional networks (FCNs)** - replace fully-connected layers with convolutional layers
  - Deeper, updated models (now ResNet) consistent with ImageNet Challenge object classification tasks.

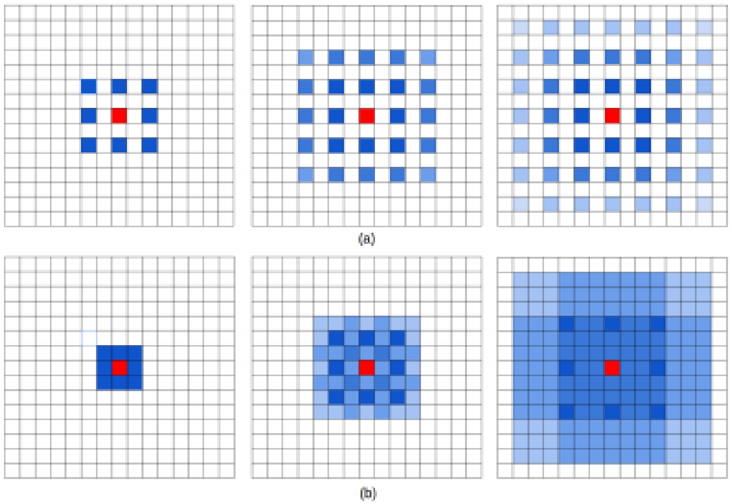
- **Conditional Random Fields (CRFs)** to capture both local and long-range dependencies within an image to refine the prediction map.

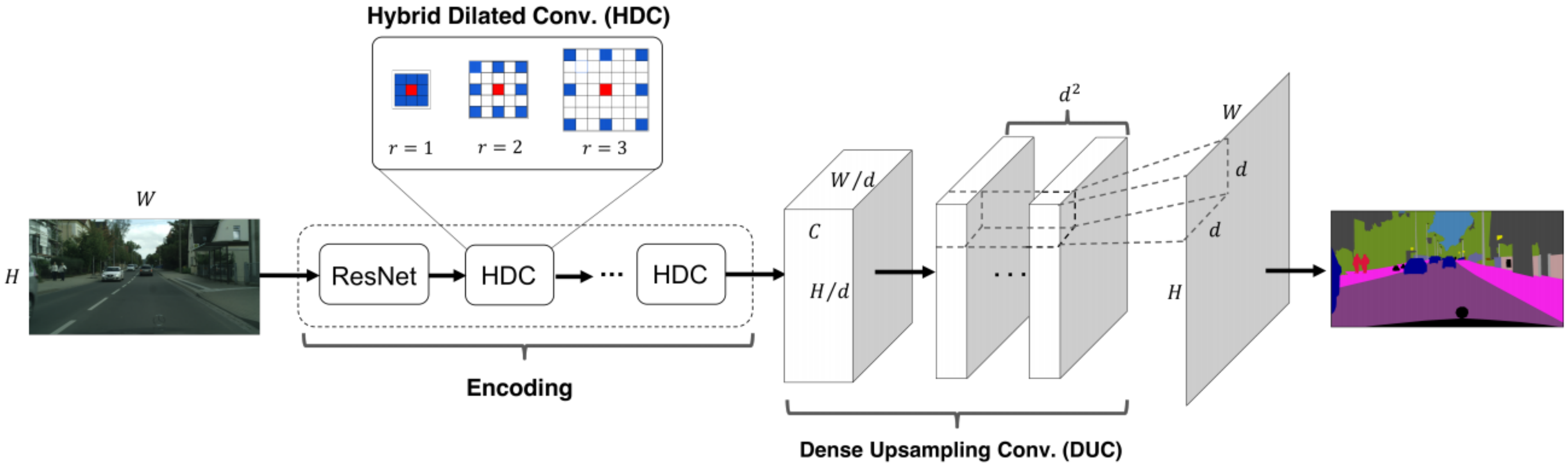
- **Dilated convolution** (aka Atrous convolution) – maintain computational cost, increase resolution of intermediate feature maps

ResNet-DUC (Nov 2017)

Paper: "Understanding Convolution for Semantic Segmentation"

- Dense upsampling convolution (DUC) instead of bilinear upsampling
  - **Learnable:** Learn the upscaling filters
- Hybrid dilated convolution (HDC)
  - Use a different dilation rate





FlowNet (May 2015)

Paper: “FlowNet: Learning Optical Flow with Convolutional Networks ”

- Learn flow from image-pair, end to end.
    - FlowNetS – stacks two images as input
    - FlowNetC – convolute separately, combine with correlation layer

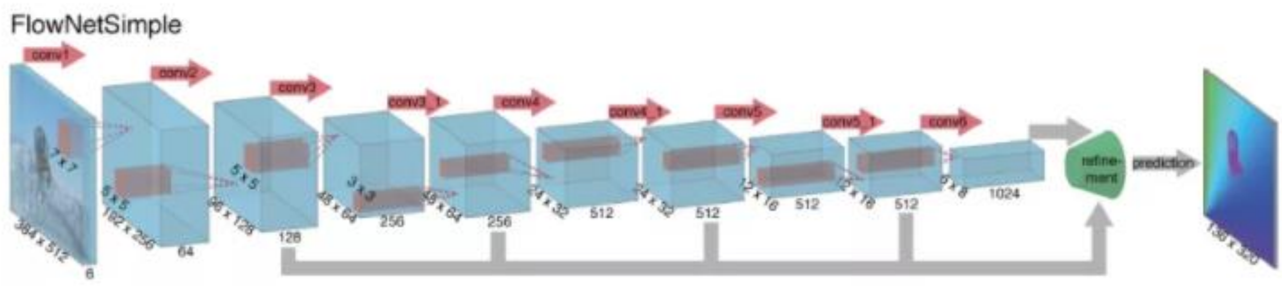
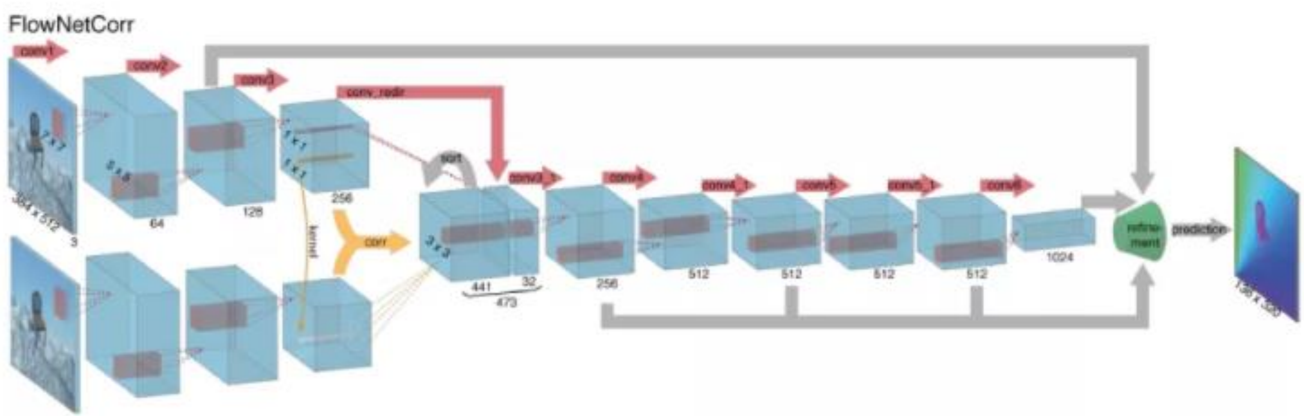


Fig. 1

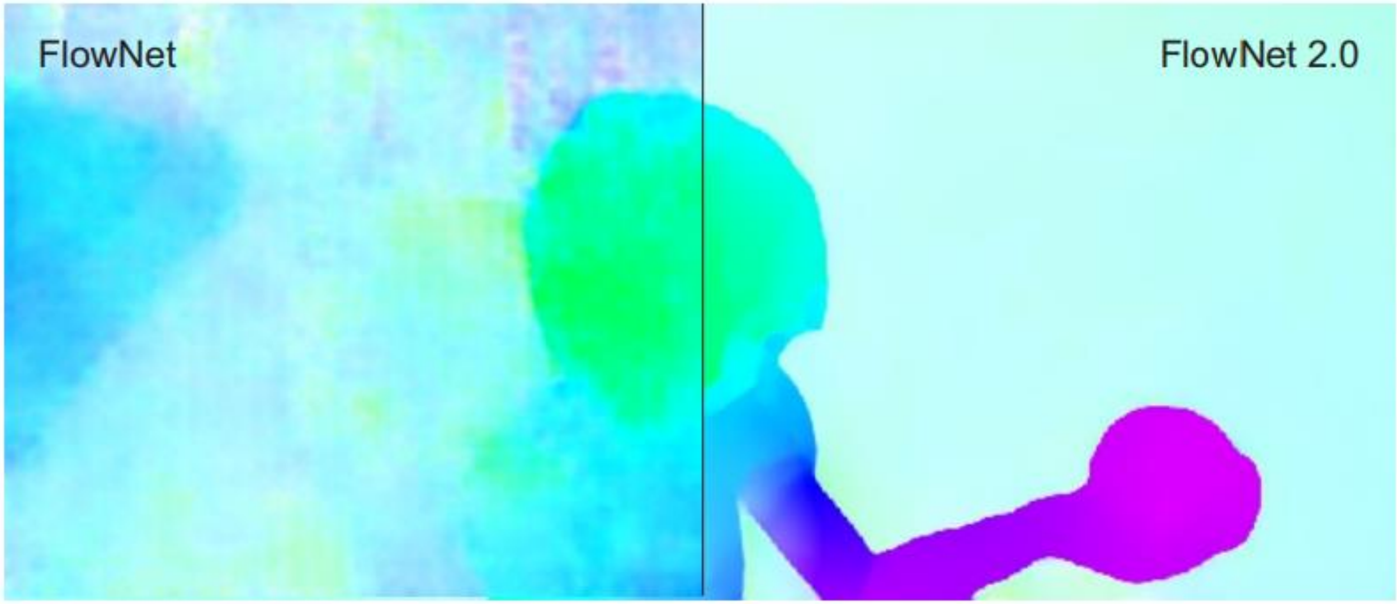


FlowNet 2.0 (Dec 2016)

Paper: "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks"

- Stack FlowNetS and FlowNetC
- Improvement over FlowNet
  - Smooth flow fields
  - Preserves fine-motion detail
  - Runs at 8-140fps

- Observations:
  - Stacking networks as an approach
  - Order of training dataset matters



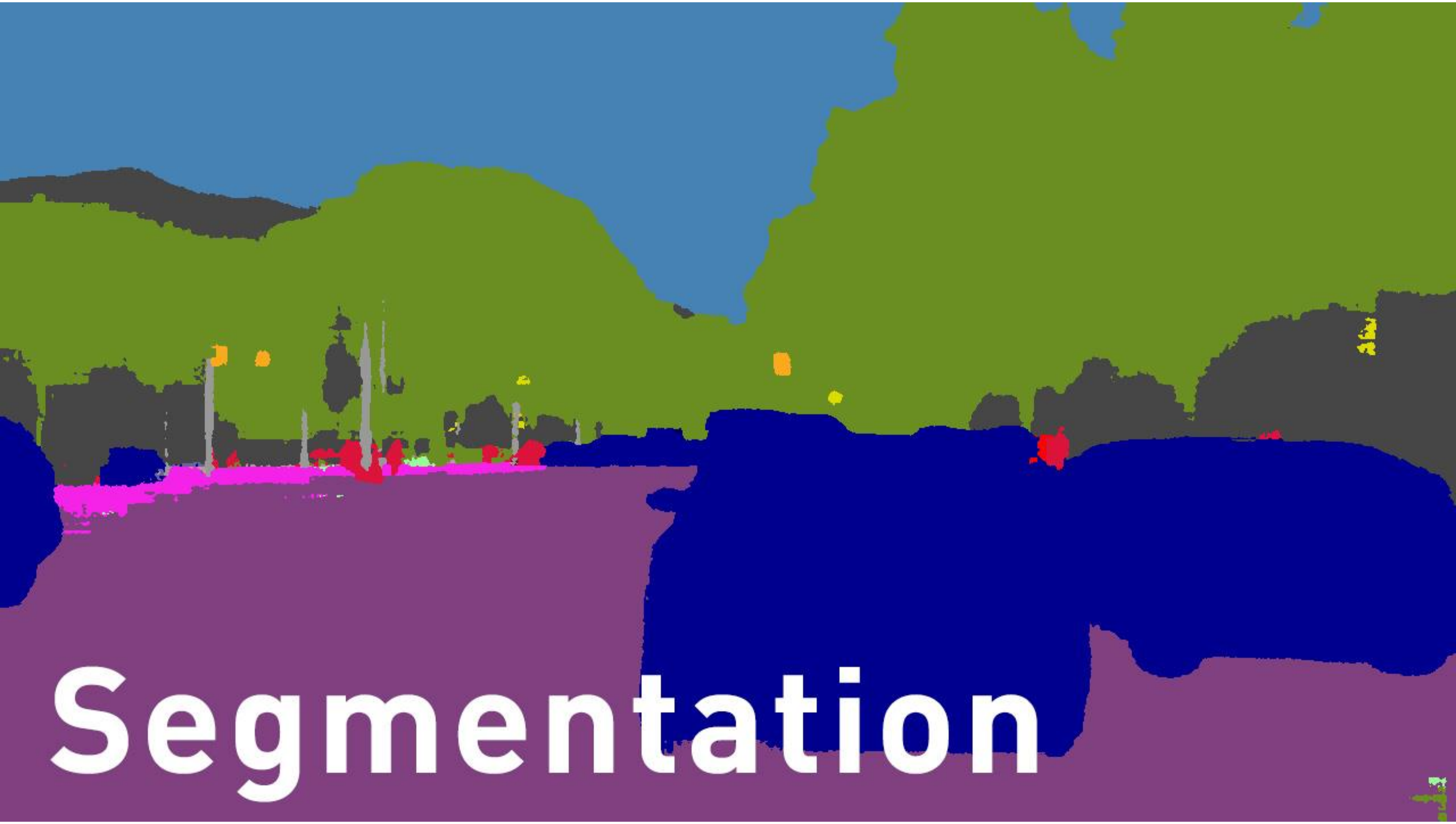
SegFuse: Dynamic Driving Scene Segmentation

cars.mit.edu/segfuse

SegFuse: Dynamic Driving Scene Segmentation

cars.mit.edu/segfuse

SegFuse: Dynamic Driving Scene Segmentation

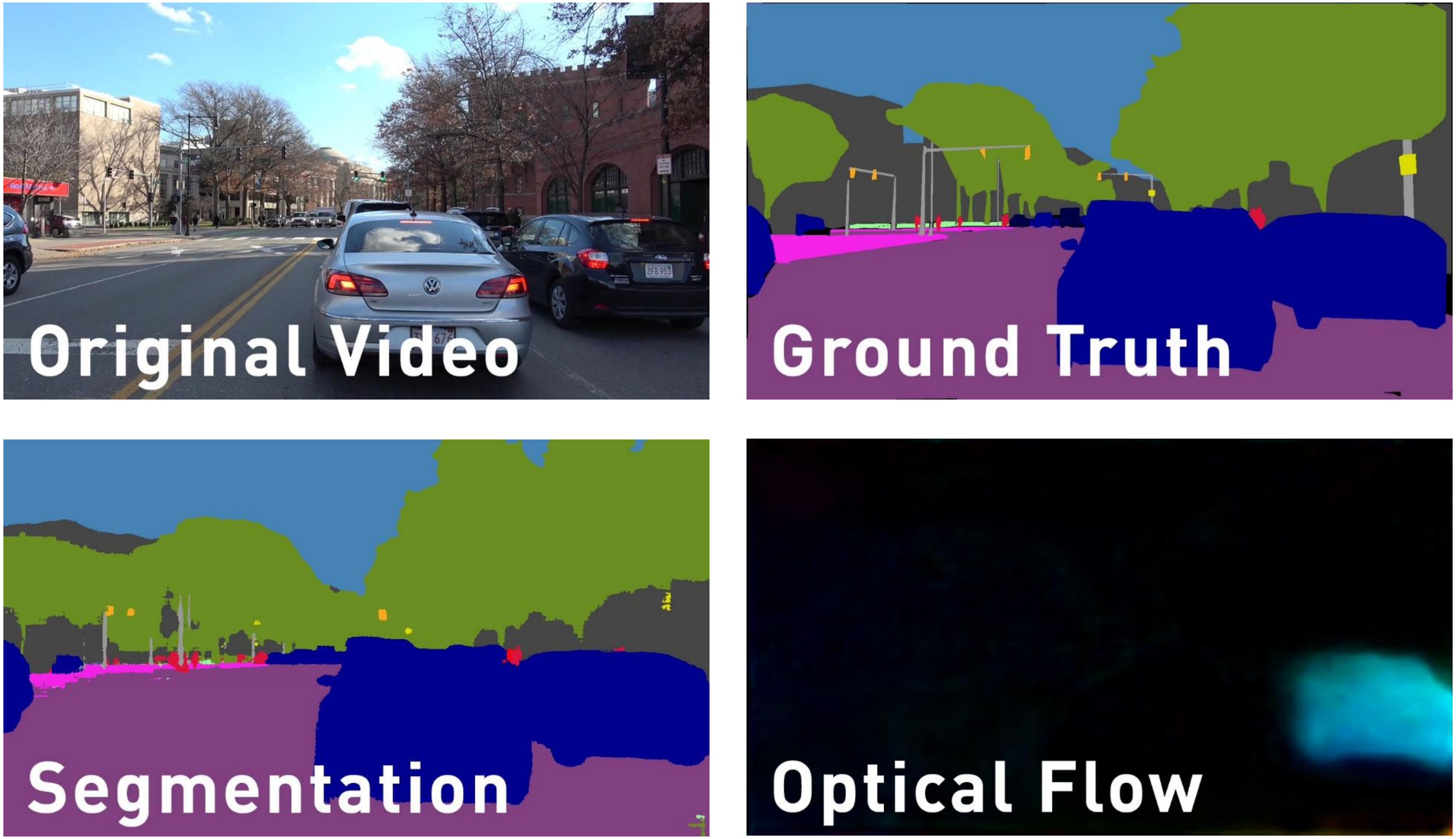


cars.mit.edu/segfuse

SegFuse: Dynamic Driving Scene Segmentation

cars.mit.edu/segfuse

SegFuse: Dynamic Driving Scene Segmentation

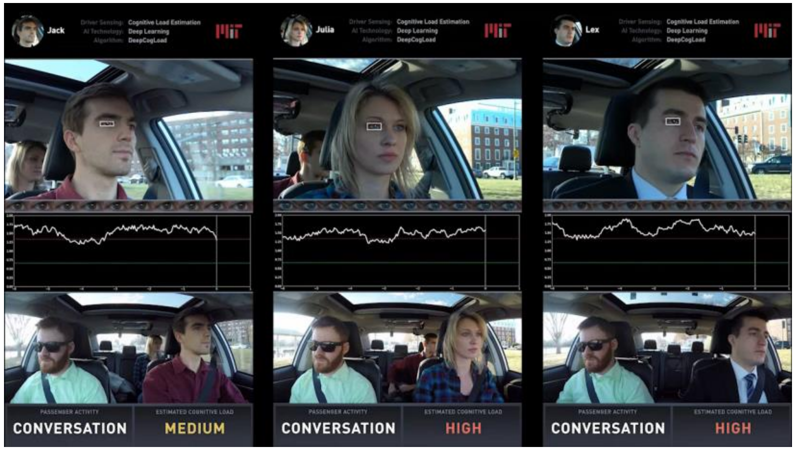


cars.mit.edu/segfuse

Thank You

Tomorrow: ***Waymo***

Next lecture: ***Deep Learning for Human Sensing***



Upcoming online-only lectures:

- Capsule networks
- Generative adversarial networks